J Control Theory Appl 2011 9 (3) 310–335
DOI 10.1007/s11768-011-1005-3

Approximate policy iteration: a survey and some new methods

Dimitri P. BERTSEKAS

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.

Abstract: We consider the classical policy iteration method of dynamic programming (DP), where approximations and simulation are used to deal with the curse of dimensionality. We survey a number of issues: convergence and rate of convergence of approximate policy evaluation methods, singularity and susceptibility to simulation noise of policy evaluation, exploration issues, constrained and enhanced policy iteration, policy oscillation and chattering, and optimistic and distributed policy iteration. Our discussion of policy evaluation is couched in general terms and aims to unify the available methods in the light of recent research developments and to compare the two main policy evaluation approaches: projected equations and temporal differences (TD), and aggregation. In the context of these approaches, we survey two different types of simulation-based algorithms: matrix inversion methods, such as least-squares temporal difference (LSTD), and iterative methods, such as least-squares policy evaluation (LSPE) and TD (λ), and their scaled variants. We discuss a recent method, based on regression and regularization, which rectifies the unreliability of LSTD for nearly singular projected Bellman equations. An iterative version of this method belongs to the LSPE class of methods and provides the connecting link between LSTD and LSPE. Our discussion of policy improvement focuses on the role of policy oscillation and its effect on performance guarantees. We illustrate that policy evaluation when done by the projected equation/TD approach may lead to policy oscillation, but when done by aggregation it does not. This implies better error bounds and more regular performance for aggregation, at the expense of some loss of generality in cost function representation capability. Hard aggregation provides the connecting link between projected equation/TD-based and aggregation-based policy evaluation, and is characterized by favorable error bounds.

Keywords: Dynamic programming; Policy iteration; Projected equation; Aggregation; Chattering; Regularization

1 Introduction

In this paper, we aim to survey and place in broad context a number of issues relating to approximate policy iteration methods for finite-state, discrete-time, stochastic dynamic programming (DP) problems. These methods are one of the major approaches for approximate DP, a field that has attracted substantial research interest and has a wide range of applications, because of its potential to address large and complex problems that may not be treatable in other ways. Among recent works in the extensive related literature, we mention textbooks and research monographs: Bertsekas and Tsitsiklis [1], Sutton and Barto [2], Gosavi [3], Cao [4], Chang, Fu, Hu, and Marcus [5], Meyn [6], Powell [7], Busoniu, Babuska, De Schutter, and Ernst [8], and the author's text in preparation [9]; edited volumes and special issues: White and Sofge [10], Si, Barto, Powell, and Wunsch [11], Lewis, Lendaris, and Liu [12], and the 2007-2009 Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning; and surveys: Barto, Bradtke, and Singh [13], Borkar [14], Lewis and Vrabie [15], and Szepesvari [16].

For an overview of policy iteration methods, let us focus on the α-discounted n-state Markovian decision problem (MDP) with states $1,\ldots,n$, controls $u \in U(i)$ at state i, transition probabilities $p_{ij}(u)$, and cost $g(i,u,j)$ for transition from i to j under control u. A (stationary) policy μ is a function from states i to admissible controls $u \in U(i)$, and $J_\mu(i)$ is the cost starting from state i and using policy μ. It is well known (see, e.g., Puterman [17] or Bertsekas [18]) that the costs $J_\mu(i)$, $i=1,\ldots,n$, are the unique solution of Bellman's equation

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \quad i=1,\ldots,n.$$

Equivalently, the vector $J_\mu \in \mathbb{R}^n$, which has components $J_\mu(i)$, [1] is the unique fixed point of the mapping $T_\mu : \mathbb{R}^n \mapsto \mathbb{R}^n$, which maps $J \in \mathbb{R}^n$ to the vector $T_\mu J \in \mathbb{R}^n$ that has components

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J(j)\big), \quad i=1,\ldots,n. \quad (1)$$

Similarly, the optimal costs starting from $i=1,\ldots,n$, are denoted $J^*(i)$ and are the unique solution of Bellman's equation

$$J^*(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad i=1,\ldots,n.$$

Received 7 January 2011.
This work was supported by the National Science Foundation (No.ECCS-0801549), the LANL Information Science and Technology Institute, and the Air Force (No.FA9550-10-1-0412).

[1] In our notation, $\mathbb{R}^n$ is the n-dimensional Euclidean space, all vectors in $\mathbb{R}^n$ are viewed as column vectors, and a prime denotes transposition. The identity matrix is denoted by I.

Equivalently, the optimal cost vector $J^* \in \mathbb{R}^n$, which has components $J^*(i)$, is the unique fixed point of the mapping $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ defined by

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \quad i = 1, \ldots, n. \quad (2)$$

An important property is that T is a sup-norm contraction, so the iteration $J^{k+1} = TJ^k$ converges to J^* from any starting point J^0 – this is known as value iteration.

Policy iteration is a major alternative to value iteration. It produces a sequence of policies and associated cost functions through iterations that have two phases: policy evaluation (where the cost function of a policy is evaluated) and policy improvement (where a new policy is generated). In the exact form of the algorithm, the current policy μ is improved by finding $\bar{\mu}$ that satisfies $T_{\bar{\mu}} J_\mu = TJ_\mu$ (i.e., by minimizing in the right-hand side of equation (2) with J_μ in place of J). The improved policy $\bar{\mu}$ is evaluated by solving the linear system of equations $J_{\bar{\mu}} = T_{\bar{\mu}} J_{\bar{\mu}}$, and $(J_{\bar{\mu}}, \bar{\mu})$ becomes the new cost vector-policy pair, which is used to start a new iteration.

In a variant of the method, the improved policy $\bar{\mu}$ is evaluated by applying $T_{\bar{\mu}}$ a finite number of times to an approximate evaluation of the preceding policy μ, which we denote by $\tilde{J}_\mu$. This is known as 'optimistic' or 'modified' policy iteration, and its motivation is that, in problems with a large number of states, the linear system $J_{\bar{\mu}} = T_{\bar{\mu}} J_{\bar{\mu}}$ cannot be practically solved directly by matrix inversion, so it is best solved iteratively by value iteration, that is, by repeated application of $T_{\bar{\mu}}$ to some initial vector (most conveniently the vector $\tilde{J}_\mu$). If the number m of applications of $T_{\bar{\mu}}$ is very large, the exact form of policy iteration is essentially obtained, but practice has shown that it is most efficient to use a moderate value of m. In this case, the algorithm looks like a hybrid of value and policy iteration, involving a sequence of alternate applications of T and T_μ, with μ changing over time.

It is well known that the exact form of policy iteration converges to an optimal cost-policy pair for any initial conditions (in a finite number of iterations when the number of policies is finite, as in the discounted MDP). The convergence of optimistic policy iteration is more fragile and requires that the initial pair $(\tilde{J}_{\mu^0}, \mu^0)$ satisfy $T_{\mu^0} \tilde{J}_{\mu^0} \leqslant \tilde{J}_{\mu^0}$, when implemented in asynchronous form (irregularly, one state at a time); see Williams and Baird [19], or Section 1.3.3 in [18]. [2] Since simulation-based policy iteration algorithms are typically asynchronous, this restriction has motivated algorithmic modifications with guaranteed convergence from arbitrary initial conditions, which will be partly discussed in Section 3.8.

1.1 Policy iteration methods with cost function approximation

In policy iteration methods with cost function approximation, we evaluate μ by approximating J_μ with a vector Φr_μ from the subspace S spanned by the columns of an $n \times s$ matrix Φ, which may be viewed as basis functions: [3]

$$S = \{\Phi r | r \in \mathbb{R}^s\}.$$

We generate an 'improved' policy $\bar{\mu}$ using the formula $T_{\bar{\mu}}(\Phi r_\mu) = T(\Phi r_\mu)$, i.e.,

$$\bar{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha \phi(j)' r_\mu), \quad i = 1, \ldots, n, \quad (3)$$

where $\phi(j)'$ is the row of Φ that corresponds to state j (the method terminates with μ if $T_\mu(\Phi r_\mu) = T(\Phi r_\mu)$). We then repeat with μ replaced by $\bar{\mu}$.

In this paper, for generality and notational simplicity, we consider MDP where for each policy μ, the mapping $T_\mu : \mathbb{R}^n \mapsto \mathbb{R}^n$ has the form

$$T_\mu J = g_\mu + A_\mu J \quad (4)$$

for given $n \times n$ matrix A_μ with nonnegative components $a_{ij}\big(\mu(i)\big)$ and vector $g_\mu \in \mathbb{R}^n$ with components $g_\mu(i)$. The α-discounted n-state MDP is the special case where $A_\mu = \alpha P_\mu$, with P_μ being the matrix of transition probabilities $p_{ij}(\mu(i))$ corresponding to μ, and g_μ the vector with ith component $g_\mu(i) = \sum_{j=1}^{n} p_{ij}(\mu(i)) g\big(i, \mu(i), j\big)$, the expected one-stage cost at state i under control $\mu(i)$. In policy iteration with cost function approximation, we compute approximations Φr_μ to J_μ using some policy evaluation method, and we use them to compute new policies based on (approximate) policy improvement, in analogy with equation (3).

In Section 2, we focus just on the approximate evaluation of a single fixed policy μ, so for notational simplicity we drop the subscript μ from equation (4). We thus focus on approximations of the form Φr for the fixed point of the linear mapping

$$TJ = g + AJ, \quad (5)$$

where A is an $n \times n$ matrix with nonnegative components a_{ij} and $g \in \mathbb{R}^n$. Several algorithmic approaches have been proposed for this purpose. The most popular are

a) Projected equation approach: Here, the approximation Φr is obtained by solving the projected equation

$$\Phi r = \Pi T(\Phi r), \quad (6)$$

where Π denotes projection onto the subspace S. The projection is with respect to a weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1, \ldots, \xi_n)$ is a probability distribution with

[2] Convergence of optimistic policy iteration is obtained for the discounted MDP, without the condition $T_{\mu^0} \tilde{J}_{\mu^0} \leqslant \tilde{J}_{\mu^0}$ (see page 88 of [18]). However, this is not so under asynchronous implementations.

[3] We focus on linear/subspace cost function approximation in this paper. Naturally, the choice of the matrix Φ is critical for effective policy evaluation. The ith row of Φ consists of s numbers, which may be viewed as features that characterize state i, and are weighted with the corresponding s components of r_μ to provide an approximation of the cost starting from i under the given policy. Features, when well crafted, can capture the dominant nonlinearities of the cost vector, and their linear combination can work well as an approximation architecture. In this paper, we do not discuss the selection of Φ, but we note the possibility of its optimal choice within some restricted class by using gradient and random search algorithms (see references [20–23] for recent work on this subject). Note that some of the methods discussed in this paper can be adapted so that they can use 'sampled' features, that is, features whose values are corrupted by stochastic noise (see Section 6 in [24]).

positive components (i.e., $\|J\|_\xi^2 = \sum_{i=1}^{n} \xi_i x_i^2$, where $\xi_i > 0$ for all i). The distribution vector ξ is sometimes (but by no means always) the steady-state probability vector of a Markov chain associated with the MDP.

b) Aggregation approach: Here, the approximation Φr is obtained by solving an equation of the form

$$r = DT(\Phi r), \tag{7}$$

where D is a matrix whose rows are probability distributions. Contrary to the projected equation approach, where Φ is arbitrary, an important restriction is that Φ must also be a matrix whose rows are probability distributions. The components of D and Φ are known as the disaggregation and aggregation probabilities. The vector r can be viewed as a cost vector of an aggregate problem that has s states and is defined by D and Φ (see Section 4 for details and interpretation).

There are also optimistic versions of these two approaches, where the corresponding equation (6) or (7) is solved approximately, with a finite number of iterations of some iterative method. Generally, the projected equation approach is associated with temporal difference (TD) methods, which originated in reinforcement learning with the works of Samuel [25, 26] on a checkers-playing program. The papers by Barto, Sutton, and Anderson [27] and Sutton [28] proposed the TD(λ) method, which motivated a lot of research in simulation-based DP, particularly following an early success with the backgammon playing program of Tesauro [29]. The original papers did not make the connection of TD methods with the projected equation. Indeed, for quite a long time, it was not clear which mathematical problem TD(λ) was aiming to solve. The mathematical convergence and other properties of TD(λ) and its connections with the projected equation were clarified in the mid-1990s through the works of several authors, including Gurvits et al. [30], Jaakkola et al. [31], Pineda [32], and Tsitsiklis and Van Roy [33, 34]. More recent works have focused on the use of least-squares-based TD methods, such as the LSTD method (Bradtke and Barto [35]) and the LSPE method (Bertsekas and Ioffe [36]), which will be discussed later.

Note that the projected equation framework has a long history in the context of Galerkin methods for the approximate solution of high-dimensional or infinite-dimensional linear equations (partial differential, integral, inverse problems, etc. See, e.g., references [37, 38]). In fact, much of the policy evaluation theory discussed in this paper applies to general projected equations arising in contexts beyond DP (see references [1, 24, 39–42]). However, the use of the Monte Carlo simulation and Monte Carlo integration ideas that are central in approximate DP is an important characteristic that differentiates the methods of the present paper from the Galerkin methodology, as currently practiced in the numerical analysis field.

The aggregation approach also has a long history in scientific computation and operations research. It was introduced in the simulation-based approximate DP context, mostly in the form of value iteration; see Singh et al. [43, 44], Gordon [45], Tsitsiklis and Van Roy [46], and Van Roy [47]. Currently, the aggregation approach seems to be less popular, but as we will argue in this paper, it has some interesting advantages over the projected equation approach, even though there are restrictions in its applicability (see also the discussion in [9]).

Let us discuss briefly some alternative policy iteration approaches. In one of them, evaluation is based on the minimization of the (squared) Bellman equation error, as expressed by

$$\min_{r\in\mathbb{R}^s} \|\Phi r - T(\Phi r)\|^2, \tag{8}$$

where T is the mapping (5). The summation over all the states in the squared norm above is usually approximated by a summation over a smaller 'representative' set of states, often obtained in part through simulation. The optimality conditions for this minimization can be shown to be equivalent to the projected equation $\Phi r = \Pi\hat{T}(\Phi r)$, where $\hat{T}$ is the mapping

$$\hat{T}J = TJ + A'(J - TJ),$$

which has the same fixed points as T, provided $I - A'$ is an invertible matrix (see Section 6.8.4 in [9]). Thus, the Bellman equation error approach is a special case of the projected equation approach, although it leads to different simulation-based implementations. Sometimes, it gives a better approximation error and sometimes worse. It has some interesting characteristics; see, e.g., the discussion in [1] (Sections 6.10 and 6.11). However, it is less efficient when implemented by simulation (it requires the simultaneous generation of two state trajectories instead of one), and it is more susceptible to simulation noise; for this reason, it is not as popular as the projected equation approach at present. We do not consider this approach in this paper, referring to the literature for further discussion (e.g., Sections 6.10 and 6.11 in [1], Section 6.8.4 in [9], and [48]).

Among other approximate policy iteration methods, let us note the ones proposed recently by Thiery and Scherrer [49] (see also [9]), which are simulation-based implementations of λ-policy iteration. This latter algorithm combines elements of value and policy iteration and was proposed by Bertsekas and Ioffe [36] in conjunction with the LSPE method (see also Section 2.3.1 in [1]). The implementations of [49] combine elements of several of the ideas that we will discuss in Sections 2 and 3, including solution of linear systems by simulation (Section 2.3), a λ parameter to control the bias-variance tradeoff (Section 2.3), convenience of exploration (Section 3.2), and optimistic operation (Section 3.3).

Contrary to approximate policy evaluation, which is an extensively researched and reasonably well-understood subject, policy improvement with cost function approximation may exhibit complex behavior that is hard to analyze and can have seriously adverse impact on practical performance. An important question relates to the issue of inadequate exploration, i.e., the potential bias of the simulation through underrepresentation of states that are unlikely to occur under the policy being simulated. Another important question is whether the policy iteration process is seriously hampered by oscillations between poor policies, roughly similar to the attraction of gradient methods to poor local minima. As we will discuss later, there has been little apparent concern in the approximate DP/reinforcement learning literature about this possibility, even though it has been documented with several simple examples and may be responsible for the poor performance of approximate policy iteration recently

demonstrated in case studies involving the game of tetris (see the discussion in Section 3.5).

Let us also note that approximate policy iteration methods are based on the idea of 'approximation in value space' and hence also on the hypothesis that a more accurate cost-to-go approximation will yield a better one-step or multistep lookahead policy through the policy improvement equation (3). This is a reasonable but by no means self-evident hypothesis and may in fact not even be true in a given problem. Competitor methods are based on the idea of 'approximation in policy space' and use simulation in conjunction with a gradient or random search method to approximate directly an optimal policy with a policy of a given parametric form. These methods do not aim at good cost function approximation through which a well-performing policy may hopefully be obtained. Rather, they aim directly at finding a policy with good performance. The circumstances under which one type of method may yield a policy with superior performance over the other are unclear at present.

In this survey paper, we first address various computational issues of policy evaluation, including convergence and rate of convergence and sensitivity to simulation noise in the case where Φ is (nearly) rank deficient (Section 2). We then discuss additional issues that arise when policy evaluation is embedded within policy iteration, such as exploration, limited sampling/optimistic policy iteration, policy oscillation, and error bounds (Section 3). For policy evaluation, we first focus on the projected equation approach, since this is the one that is better known and has the most complex and multifaceted behavior. We return to policy evaluation using the aggregation approach in Section 4. However, we take special care to introduce sufficient generality and abstraction into our presentation of Sections 2 and 3, so that a coherent and unified view of the commonalities and differences of the two approaches emerges.

Since we cover a lot of ground, to keep the size of the paper reasonable, we adopt a somewhat abstract and research-oriented style of exposition. We give proofs of some results, we refer to the published literature for others, and we speculate on a few other results that are plausible but not yet rigorously established.

2 Projected equation methods for policy evaluation

In this section, we discuss general aspects of policy evaluation using the projected equation

$$\Phi r = \Pi T(\Phi r) = \Pi(g + A\Phi r),$$

cf. equations (5) and (6), and we give the forms of the two basic methods: matrix inversion and iterative. While these methods apply to general Galerkin approximation, we focus primarily on TD-like, simulation-based algorithms that were developed in approximate DP. For frameworks that are based on a more general viewpoint, we refer to our work on the solution of linear systems of equations [1, 24, 40], variational inequalities [39], and least-squares and inverse problems [50]. One of the advantages of using a general matrix A in the definition of T is that it allows simultaneous application of our results to all types of Bellman equations (e.g., discounted, stochastic shortest path, average cost, semi-Markov, single-step, and multistep).

We recall the basic characterization of the projection: $\Pi T(J)$ is the unique vector $Y \in S$ such that $Y - TJ$ is orthogonal to all vectors of S in the scaled geometry of the inner product $\langle x_1, x_2 \rangle = x_1' \Xi x_2$, where Ξ is the diagonal matrix with the components of the probability distribution vector $\xi = (\xi_1, \ldots, \xi_n)$ on the diagonal. Since the projected equation is to find $J \in S$ such that TJ projects onto J, its solutions are characterized by the orthogonality principle: J is a solution if and only if $J \in S$ and $J - TJ$ is orthogonal to all the columns of Φ, which span the subspace S, i.e., $\Phi' \Xi (J - TJ) = 0$. Thus, a vector J solves the projected equation $J = \Pi T(J)$ if and only if it is of the form Φr and

$$\begin{aligned} 0 &= \Phi' \Xi (J - TJ) = \Phi' \Xi \big(\Phi r - T(\Phi r)\big) \\ &= \Phi' \Xi \big(\Phi r - (g + A\Phi r)\big). \end{aligned}$$

It follows that solving the projected equation is equivalent to solving the linear system $Cr = d$, where [4]

$$C = \Phi' \Xi (I - A)\Phi, \quad d = \Phi' \Xi g. \tag{9}$$

In this paper, when referring to the projected equation, we make no distinction between the two equivalent forms $Cr = d$ and $J = \Pi T(J)$. The set of solutions of $J = \Pi T(J)$ is obtained from the set of solutions of $Cr = d$ by multiplication with Φ. Note that if Φ does not have rank s, C will not be invertible, and $Cr = d$ will not have a unique solution even if $J = \Pi T(J)$ has a unique solution (equivalently, $I - \Pi A$ is invertible). Moreover, solutions of $J = \Pi T(J)$ depend only on the projection norm and the subspace S, and not on the matrix Φ, which provides just an algebraic representation of S. Thus, if $J = \Pi T(J)$ has a unique solution $\tilde{J}$, we have $\Phi r^* = \tilde{J}$ for every matrix Φ whose range space is S and solution r^* of the corresponding equation $Cr = d$.

2.1 Contraction properties of the projected equation

The projected equation $Cr = d$ may in principle be solved by matrix inversion, $r^* = C^{-1}d$. This does not require any contraction properties for the operators T or ΠT; invertibility of C is the only requirement. The equation may also be solved by one of several possible iterative methods (an important example is the fixed point iteration $\Phi r_{k+1} = \Pi T(\Phi r_k)$). For some of these methods, ΠT must be a contraction in order to guarantee convergence. However, even if T is a contraction with respect to some norm, ΠT need not be a contraction, unless Π is nonexpansive with respect to that norm, and this is a major source of complications for iterative methods. The following proposition provides a condition, which guarantees that ΠT is a contraction.

Proposition 1 Let $\xi = (\xi_1, \ldots, \xi_n)$ be the steady-state probability vector of an irreducible transition probability matrix Q. Then, T and ΠT are contraction mappings with respect to $\|\cdot\|_\xi$ if $a_{ij} \leqslant q_{ij}$ for all (i, j), and there exists an index $\bar{i}$ such that $a_{\bar{i}j} < q_{\bar{i}j}$ for all $j = 1, \ldots, n$.

The proof of the proposition is given in [24], and is based

[4] This line of development of the projected equation can be generalized to the case where Π is the operator that projects on a convex strict subset of S. In this case, the projected equation is equivalent to a variational inequality (in place of $Cr = d$), thereby leading to a broad extension of the overall approximation approach; see [39] and the discussion in Section 3.4.

on the inequality

$$\|QJ\|_\xi \leqslant \|J\|_\xi, \ \ \forall J \in \mathbb{R}^n,$$

which holds for any irreducible transition matrix Q with invariant distribution ξ (see, e.g., Lemma 6.4 in [1], or Lemma 6.3.1 in [18]). In a discounted MDP case where $A = \alpha P$ and P is irreducible, by taking $Q = P$, the inequality implies that T is a contraction with respect to $\|\cdot\|_\xi$ of modulus α, which together with the nonexpansiveness of Π with respect to $\|\cdot\|_\xi$, implies that the same is true for ΠT. The more general condition $a_{\bar{i}\bar{j}} < q_{\bar{i}\bar{j}}$, $j = 1, \ldots, n$, is satisfied in discounted MDP, where Q is a somewhat modified version of P to allow for exploration (see [9]), and in a stochastic shortest path MDP, where A is a substochastic matrix corresponding to a proper policy (one for which the termination state is reached with probability 1 from any starting state); see [9, 18]. There are also other related conditions that guarantee that ΠT is a contraction and apply among others to average cost MDP (see [24, 51], or [9]).

If ΠT is a contraction, then the projected equation $J = \Pi T(J)$ has a unique solution Φr, which implies that if Φ has rank s, then $Cr = d$ has a unique solution and so C is invertible. This and other properties are collected in the following proposition.

Proposition 2 Let ΠT be a contraction with respect to $\|\cdot\|_\xi$ and let Φ have rank s.

a) C is invertible and positive definite in the sense that

$$r'Cr > 0, \ \ \forall r \neq 0.$$

b) C has eigenvalues with positive real parts.

c) The unique solution r^* of the projected equation $Cr = d$ and any solution J^* of the original equation $J = TJ$ satisfy

$$\|J^* - \Phi r^*\|_\xi \leqslant \frac{1}{\sqrt{1-\alpha^2}}\|J^* - \Pi J^*\|_\xi, \qquad (10)$$

where α is the modulus of contraction of ΠT.

The results of the proposition are well known in the theory of TD methods and have been given in various forms by Tsitsiklis and Van Roy [33, 34]. Proofs can also be found in [9]. The error bound (10) is quite conservative (usually in MDP α is very close to 1). There are sharper error bounds, due to Yu and Bertsekas [40] (see also Scherrer [48]), which depend on the finer structure of A, Ξ, and the subspace S (not just on the contraction modulus of ΠT), and also apply to the case where ΠT is not a contraction.

2.2 Deterministic iterative methods for projected equations

We will now discuss iterative methods as alternatives to matrix inversion for solving the projected equation $Cr = d$. We first discuss deterministic methods that do not use simulation. These methods are not practical when the number of states is large, since they require high-dimensional matrix-vector computations. However, we will subsequently use Monte Carlo integration ideas to construct simulation-based versions that require low-dimensional calculations only.

We focus on iterative methods of the form

$$r_{k+1} = r_k - \gamma G(Cr_k - d), \qquad (11)$$

where G is a scaling matrix and γ is a positive stepsize. [5] This iteration is convergent if $I - \gamma GC$ is a contraction with respect to some norm (equivalently has all its eigenvalues strictly within the unit circle), but (under certain conditions) may also converge even if $I - \gamma GC$ is just nonexpansive with respect to some norm, which may happen if C is singular. Some interesting special choices of γ and G are given in the following three propositions. The last two apply to cases where C is singular.

Proposition 3 If ΠT is a contraction with respect to $\|\cdot\|_\xi$ and Φ has rank s, the iteration

$$r_{k+1} = r_k - (\Phi'\Xi\Phi)^{-1}(Cr_k - d) \qquad (12)$$

is equivalent to the fixed point iteration $J_{k+1} = \Pi T(J_k)$ and converges to the unique solution of the projected equation $Cr = d$.

Proof We use the projection formula

$$\Pi = \Phi(\Phi'\Xi\Phi)^{-1}\Phi'\Xi,$$

which is valid when Φ has rank s, together with the definition (9) of C and d, to write the iteration (12) as $J_{k+1} = \Pi T(J_k)$, where $J_k = \Phi r_k$, so convergence follows from the contraction property of ΠT.

Proposition 4 If ΠT is a contraction with respect to $\|\cdot\|_\xi$ and G is positive definite symmetric, there exists $\bar{\gamma} > 0$ such that for all $\gamma \in (0, \bar{\gamma}]$, the iteration

$$r_{k+1} = r_k - \gamma G(Cr_k - d)$$

converges to some solution of the projected equation.

Proof We prove the result for the case where Φ has rank s. The proof for the general case is more complicated and is based on the monotonicity property of the mapping $I - T$ (a consequence of the contraction property of ΠT) and the theory of projection methods for monotone variational inequalities (see [39, 54]).

If G is symmetric and positive definite, the matrix $G^{\frac{1}{2}}$ exists and is symmetric and positive definite. Let $M = G^{\frac{1}{2}}CG^{\frac{1}{2}}$, and note that since C is positive definite, M is also positive definite, so from Proposition 2 b) it follows that its eigenvalues have positive real parts. The eigenvalues of M and GC are equal (with eigenvectors that are multiples of $G^{\frac{1}{2}}$ or $G^{-\frac{1}{2}}$ of each other), so the eigenvalues of GC have positive real parts. It follows that the eigenvalues of $I - \gamma GC$ lie strictly within the unit circle for sufficiently small $\gamma > 0$.

Proposition 5 The iteration $r_{k+1} = r_k - G(Cr_k - d)$, where

$$G = (C'\Sigma^{-1}C + \beta I)^{-1}C'\Sigma^{-1},$$

Σ is a positive definite symmetric matrix, and β is a positive scalar, converges to some solution of the projected equation, assuming at least one solution exists.

[5] Iterative methods that involve incremental changes along directions of the form $Gf(r)$ are common and fundamental for solving a system of equations $f(r) = 0$. They arise prominently in cases where $f(r)$ is the gradient of a cost function, or the case where the equation $f(r) = 0$ can be viewed as a monotone variational inequality (cf. Proposition 2 a); see Section 3.5.3 in [52]). The iterative methods (11), with G being positive definite symmetric, coincide with the class of projection methods for monotone variational inequalities; see [39], which adopts a variational inequality view for the projected equation in a more general context that allows projection on a convex subset of the subspace S. The paper [39] also interprets the scaling matrix G in terms of 'feature scaling' (a representation of the subspace S in a different coordinate system). See also Yao and Liu [53] for some algorithms related to (11), but developed from a different viewpoint.

Proof This follows from standard convergence results about the proximal point algorithm.[6] We give a proof for the easier case where C is invertible, so the projected equation has the unique solution $r^* = C^{-1}d$. Assuming C is invertible, the matrix $C'\Sigma^{-1}C$ is positive definite, so its eigenvalues $\lambda_1, \ldots, \lambda_s$ are positive. Let $U\Lambda U'$ be the singular value decomposition of $C'\Sigma^{-1}C$, where $\Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_s\}$ and U is a unitary matrix ($UU' = I$; see [58,59]). We also have $C'\Sigma^{-1}C + \beta I = U(\Lambda + \beta I)U'$, so

$$GC = \big(U(\Lambda + \beta I)U'\big)^{-1} U\Lambda U' = U(\Lambda + \beta I)^{-1}\Lambda U'.$$

It follows that the eigenvalues of GC are $\lambda_i/(\lambda_i + \beta)$, $i = 1, \ldots, s$, and lie in the interval $(0, 1)$, so the eigenvalues of $I - GC$ also lie in the interval $(0, 1)$.

Generally, the algorithms of the preceding three propositions converge linearly, at a rate that depends strongly on the choices of γ and G. However, we will argue later that, when implemented by simulation, they all converge at the same asymptotic rate, which is dictated by the slow convergence rate of the simulation. Note that the algorithms converge under different conditions and offer alternative advantages. In particular, the algorithm of Proposition 5 is the most generally applicable, as it does not require that ΠT is a contraction or that C is invertible and, moreover, does not require a stepsize choice. At the other extreme, the simplest algorithm is

$$r_{k+1} = r_k - \gamma(Cr_k - d), \tag{13}$$

the special case of Proposition 4 that corresponds to $G = I$. However, this algorithm requires that ΠT be a contraction and a choice of stepsize from within a range that is usually unknown a priori. The iteration (12) of Proposition 3 may be viewed as intermediate between the preceding two in terms of applicability and overhead per iteration.

2.3 Iterative methods using simulation-based approximations

Unfortunately, for problems of very high dimension n, it is difficult to calculate C and d explicitly, because of the high-dimensional inner products involved in their definition (9). An alternative that has a long history in approximate DP (see the textbook references in Section 1) is to calculate simulation-based estimates $\hat{C}$ and $\hat{d}$ of C and d, respectively, based on a number of samples, and obtain an approximate solution

$$\hat{r} = \hat{C}^{-1}\hat{d}$$

by matrix inversion, assuming the inverse exists (this is the LSTD method, first proposed by Bradtke and Barto [35] and followed up by Boyan [60] and Nedić and Bertsekas [61]).[7]

An alternative to LSTD is to solve by iteration a simulation-based approximation to the projected equation $Cr = d$: we approximate the deterministic iteration

$$r_{k+1} = r_k - \gamma G(Cr_k - d),$$

(11) with

$$r_{k+1} = r_k - \gamma\hat{G}(\hat{C}r_k - \hat{d}), \tag{14}$$

where $\hat{C}$ and $\hat{d}$ are simulation-based estimates of C and d, γ is a positive stepsize, and $\hat{G}$ is an $s \times s$ matrix, which may also be obtained by simulation. Assuming that $I - \gamma\hat{G}\hat{C}$ is a contraction, this iteration will yield a solution to the system $\hat{C}r = \hat{d}$, which will serve as a simulation-based approximation to a solution of the projected equation $Cr = d$. Chronologically, the first method of this type is LSPE [36], where $\gamma = 1$ and $\hat{G}$ is an approximation to $(\Phi'\Xi\Phi)^{-1}$ (cf. equation (12)). The general method (14) may be viewed as a scaled version of LSPE, obtained by multiplying the LSPE direction with a suitable scaling matrix and by introducing a stepsize γ (see [39]).

Like LSTD, the iteration (14) may be viewed as a batch simulation approach: we first simulate to obtain $\hat{C}$, $\hat{d}$, and $\hat{G}$ and then solve the system $\hat{C}r = \hat{d}$ by the iterative method (14) rather than direct matrix inversion. An alternative is to iteratively update r as simulation samples are collected and used to form ever improving approximations to C and d. In particular, one or more iterations of the form (14) may be performed between collections of additional batches of simulation samples to improve the accuracy of $\hat{C}$ and $\hat{d}$. In the most extreme type of such an algorithm, the iteration (14) is used after a single new sample is collected. In this case, we approximate the iteration (11) by

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k), \tag{15}$$

where G_k, C_k, and d_k are simulation-based approximations, which are updated after a single new simulation sample is collected. For the purposes of further discussion, we will focus on this type of algorithm, with the understanding that there are related versions that use forms of batch simulation between iterations of the form (15) and have similar properties.

The convergence properties of iteration (15) under the condition

$$C_k \to C, \quad d_k \to d, \quad G_k \to G$$

can be for the most part inferred from the results of Propositions 3–5. In particular, if ΠT is a contraction and Φ has rank s, the proof of Proposition 3 carries through when G_k converges to $(\Phi'\Xi\Phi)^{-1}$. Similarly, if ΠT is a contraction, C is invertible, and G_k converges to a positive definite symmetric matrix, the proof of Proposition 4 carries through. Also, if C is invertible and

$$G_k = (C_k'\Sigma_k^{-1}C_k + \beta I)^{-1}C_k'\Sigma_k^{-1},$$

where $C_k \to C$, and $\Sigma_k \to \Sigma$ for a symmetric positive definite matrix Σ, the proof of Proposition 5 carries through (a formal convergence analysis for the case where C is singular is not available at present, even though the deterministic version of the iteration is convergent as per Proposition 5).

We now discuss a simulation mechanism for estimating

[6] One may write the iteration as

$$r_{k+1} = \arg\min_{r\in\mathbb{R}^s}\big[(Cr - d)'\Sigma^{-1}(Cr - d) + \beta\|r - r_k\|^2\big],$$

which is the proximal point algorithm applied to the problem of minimizing the cost function $(Cr - d)'\Sigma^{-1}(Cr - d)$. Then, Proposition 5 follows from the results of Martinet [55] and Rockafellar [56]; see also Proposition 6.5.1 in [57]. An additional result is that $\{r_k\}$ will converge to a minimizing point of $(Cr - d)'\Sigma^{-1}(Cr - d)$ even if the projected equation $Cr = d$ has no solution.

[7] Estimating C and d by simulation is motivated by the advantage that Monte Carlo summation (or integration) holds in similar circumstances: using sampling to estimate sums of a very large number of terms (or complicated integrals, respectively), with computational complexity that depends on the 'variance' of the terms added, and not on the number of terms.

C and d, proposed in [24]. It is similar to methods used traditionally in TD methods for policy evaluation, which are based on sampling the Markov chain corresponding to the policy. However, our simulation mechanism may use two Markov chains and is well suited for sampling with enhanced exploration (see the discussion in Section 3.2). We write C and d as expected values with respect to ξ:

$$C=\sum_{i=1}^{n}\xi_i\phi(i)\big(\phi(i)-\sum_{j=1}^{n}a_{ij}\phi(j)\big)',\quad d=\sum_{i=1}^{n}\xi_i\phi(i)g(i), \tag{16}$$

where ξ_i, $g(i)$, and a_{ij} are the components of ξ, g, and A, respectively, and $\phi(i)$ denotes the column of Φ' that corresponds to state i ($\phi(i)'$ is the ith row of Φ). We generate a sequence of indices $\{i_0,i_1,\ldots\}$ and a sequence of transitions $\{(i_0,j_0),(i_1,j_1),\ldots\}$. We use any probabilistic mechanism for this, subject to the following two requirements (cf. Fig. 1):

Row sampling The sequence $\{i_0,i_1,\ldots\}$ is generated according to the distribution ξ, which defines the projection norm $\|\cdot\|_\xi$, in the sense that with probability 1,

$$\lim_{k\to\infty}\frac{\sum_{t=0}^{k}\delta(i_t=i)}{k+1}=\xi_i,\quad i=1,\ldots,n, \tag{17}$$

where $\delta(\cdot)$ denotes the indicator function ($\delta(E)=1$ if the event E has occurred and $\delta(E)=0$ otherwise). Row sampling requires that each state i is generated with a relative frequency ξ_i specified by the projection norm $\|\cdot\|_\xi$. We will discuss later (Section 3.2) how one may design row sampling schemes corresponding to values ξ_i that adequately address the issue of exploration.

Column sampling The sequence $\{(i_0,j_0),(i_1,j_1),\ldots\}$ is generated according to a certain stochastic matrix L with transition probabilities ℓ_{ij} in the sense that with probability 1,

$$\lim_{k\to\infty}\frac{\sum_{t=0}^{k}\delta(i_t=i,j_t=j)}{\sum_{t=0}^{k}\delta(i_t=i)}=\ell_{ij},\quad i,j=1,\ldots,n. \tag{18}$$

We require that

$$\ell_{ij}>0\ \text{ if }\ a_{ij}>0, \tag{19}$$

so transitions (i,j) with $a_{ij}>0$ are generated with positive relative frequency. Other than this requirement, ℓ_{ij} need not be related to a_{ij}. For example, in an α-discounted problem, the column sampling probabilities need not be equal to the transition probabilities of the policy under evaluation. This turns out to be important because it allows exploration enhancement through the choice of the column sampling probabilities ℓ_{ij} (see the subsequent discussion of λ-methods).

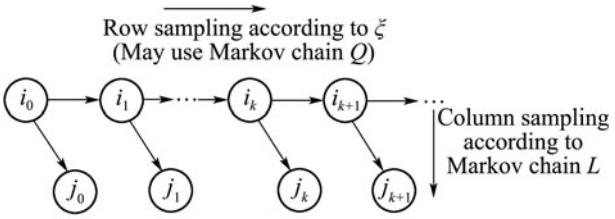

Fig. 1 The Markov chain-based simulation methodology consists of a) generating a sequence of indices $\{i_0,i_1,\ldots\}$ according to the distribution ξ (a suitable Markov chain Q may be used for this, but this is not a requirement), and b) generating a sequence of transitions $\{(i_0,j_0),(i_1,j_1),\ldots\}$ using a Markov chain L. It is possible that $j_k=i_{k+1}$, but this is not necessary. Moreover, even when A is related to the transition probability matrix of a Markov chain of an underlying MDP, Q and L need not be equal to that matrix.

At time k, we approximate C and d with

$$\begin{cases} C_k=\dfrac{1}{k+1}\displaystyle\sum_{t=0}^{k}\phi(i_t)\Big(\phi(i_t)-\frac{a_{i_tj_t}}{\ell_{i_tj_t}}\phi(j_t)\Big)', \\ d_k=\dfrac{1}{k+1}\displaystyle\sum_{t=0}^{k}\phi(i_t)g(i_t). \end{cases} \tag{20}$$

To show that this is a valid approximation, we count the number of times an index occurs, and after collecting terms, we write (20) as

$$\begin{cases} C_k=\displaystyle\sum_{i=1}^{n}\hat\xi_{i,k}\phi(i)\Big(\phi(i)-\sum_{j=1}^{n}\hat\ell_{ij,k}\frac{a_{ij}}{\ell_{ij}}\phi(j)\Big)', \\ d_k=\displaystyle\sum_{i=1}^{n}\hat\xi_{i,k}\phi(i)g(i), \end{cases} \tag{21}$$

where

$$\hat\xi_{i,k}=\frac{\sum_{t=0}^{k}\delta(i_t=i)}{k+1},\quad \hat\ell_{ij,k}=\frac{\sum_{t=0}^{k}\delta(i_t=i,j_t=j)}{\sum_{t=0}^{k}\delta(i_t=i)}.$$

In view of the assumption

$$\hat\xi_{i,k}\to\xi_i,\quad \hat\ell_{ij,k}\to\ell_{ij},\quad i,j=1,\ldots,n,$$

cf. equations (17) and (18), by comparing equations (16) and (21), we see that $C_k\to C$ and $d_k\to d$.[8]

In the iterative method

$$r_{k+1}=r_k-\gamma G_k(C_kr_k-d_k),$$

(cf. equation (15)), in addition to C_k, d_k, one may simultaneously obtain G_k with the preceding simulation mechanism. In particular, one may choose $\gamma=1$ and

$$G_k=\Big(\frac{1}{k+1}\sum_{t=0}^{k}\phi(i_t)\phi(i_t)'\Big)^{-1}, \tag{25}$$

[8] In the discounted case where $A=\alpha P$, the approximate projected equation $C_kr=d_k$ can be written as

$$C_kr-d_k=\sum_{t=0}^{k}\phi(i_t)q_{k,t}=0, \tag{22}$$

where

$$q_{k,t}=\phi(i_t)'r_k-\alpha\phi(i_{t+1})'r_k-g(i_t). \tag{23}$$

The scalar $q_{k,t}$ is the so-called TD, associated with r_k and transition (i_t,i_{t+1}). It may be viewed as a sample of a residual term arising in the projected equation. More specifically, we have

$$Cr_k-d=\Phi'\Xi(\Phi r_k-\alpha P\Phi r_k-g). \tag{24}$$

The three terms in the definition (23) of the TD $q_{k,t}$ can be viewed as samples (associated with the transition (i_t,i_{t+1})) of the corresponding three terms in the expression $\Xi(\Phi r_k-\alpha P\Phi r_k-g)$ in (24). For the purposes of this paper, TDs and the form of the approximate projected equation (22) play no substantive role, and they will not be used in any way.

which is a simulation-based approximation to $(\Phi'\Xi\Phi)^{-1}$ (cf. the iteration (12) of Proposition 3). This is the LSPE method first proposed by Bertsekas and Ioffe [36] and followed up by Nedić and Bertsekas [61], Bertsekas et al. [62], and Yu and Bertsekas [51] (the method was also described in [36] and [1] as an implementation of the λ-policy iteration method and was used to address a challenging tetris application; the name LSPE was introduced in [61]).

The LSTD and LSPE methods were preceded and motivated by the TD(0) method, which for an α-discounted problem has the form

$$r_{k+1} = r_k - \gamma_k \phi(i_k) q_{k,k}, \tag{26}$$

where $q_{k,k}$ is the TD given by (23), and γ_k is a diminishing stepsize that diminishes at an appropriate rate, such as $\gamma_k = \mathrm{O}(1/k)$. Thus, in view of (22), TD(0) has a similar form to the simple iteration (13), but differs in two important ways: first, the stepsize γ_k is diminishing, and second, C_k and d_k are single-sample estimates of C and d (only the last simulation sample is used rather than all of the k samples). TD(0) may be viewed as a stochastic approximation/Robbins-Monro scheme for solving the projected equation $Cr = d$, and in this sense, it is fundamentally different than the LSTD and LSTE methods, which are Monte Carlo versions of matrix inversion and iterative deterministic methods, which use simulation-based estimates in place of the hard-to-compute quantities C and d. Thus, TD(0) requires an entirely different type of mathematical convergence analysis than LSTD and LSPE (see, e.g., [33]). Let us also note a scaled form of TD(0), which uses a positive definite symmetric scaling matrix G in (26) and has been proposed and analyzed by Choi and Van Roy [63]. Similar to TD(0), their method uses single-sample estimates of C and d in the algorithm of Proposition 4.

Note that there is a lot of flexibility for row and column sampling to satisfy (17) and (18). For example, to satisfy (17), the indices i_t do not need to be sampled independently according to ξ. Instead, it may be convenient to introduce an irreducible Markov chain with transition matrix Q, states $1, \ldots, n$, and ξ as its steady-state probability vector and to start at some state i_0 and generate the sequence $\{i_0, i_1, \ldots\}$ as a single infinitely long trajectory of the chain. For the transition sequence, we may optionally let $j_k = i_{k+1}$ for all k, in which case L would be identical to Q, but in general this is not essential. In the MDP literature, the most common choice is $Q = L = P$, where P is the transition matrix of the associated Markov chain. However, in the context of policy iteration, it soon became clear that it is important to do row sampling using a different Markov chain, one that generates by simulation all states with sufficient frequency (see the discussion of exploration in Section 3.2).

Note also that multiple simulated sequences can be used to form the simulation estimates (20) of C and d. For example, in the Markov chain-based sampling schemes, we can generate multiple infinitely long trajectories of the chain, starting at several different states. This will work even if the chain has multiple recurrent classes, as long as there are no transient states and at least one trajectory is started from within each recurrent class. Again, ξ will be a steady-state probability vector of the chain and need not be known explicitly. Note also that using multiple trajectories may be interesting even if there is a single recurrent class, for at least two reasons:

a) The generation of trajectories may be parallelized among multiple processors, resulting in significant speedup.

b) The empirical frequencies of occurrence of the states may approach the steady-state probabilities more quickly; this is particularly so for large and 'stiff' Markov chains.

We finally note that there are several variants of the Markov chain-based simulation method outlined above (see [1] or more details). For example, if instead of $g(i_t)$ we obtain the sample $g(i_t, j_t)$, the vector d_k in (20) should be

$$d_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) g(i_t, j_t).$$

As another example, zero mean noise with appropriate independence properties may be added to a_{ij} and $g(i)$. In this case, the estimates of C and d are given by

$$\begin{cases} C_k = \dfrac{1}{k+1} \displaystyle\sum_{t=0}^{k} \phi(i_t) \Big(\phi(i_t) - \sum_{j=1}^{n} (a_{i_t j} + \zeta_t(j)) \phi(j) \Big)', \\ d_k = \dfrac{1}{k+1} \displaystyle\sum_{t=0}^{k} \phi(i_t) \big(g(i_t) + \theta_t \big), \end{cases} \tag{27}$$

where for each j, $\zeta_t(j)$ is a sequence of random variables such that, with probability 1,

$$\lim_{k\to\infty} \frac{\displaystyle\sum_{t=0}^{k} \delta(i_t = i) \zeta_t(j)}{\displaystyle\sum_{t=0}^{k} \delta(i_t = i)} = 0, \quad \forall i, j = 1, \ldots, n, \tag{28}$$

and θ_t is a sequence of random variables such that, with probability 1,

$$\lim_{k\to\infty} \frac{\displaystyle\sum_{t=0}^{k} \delta(i_t = i) \theta_t}{\displaystyle\sum_{t=0}^{k} \delta(i_t = i)} = 0, \quad \forall i = 1, \ldots, n. \tag{29}$$

This variant can be used in situations where the components a_{ij} and $g(i)$ represent the expected values of random variables whose samples can be conveniently simulated with additive 'noises' $\zeta_t(j)$ and θ_t, respectively, such that equations (28) and (29) hold with probability 1.

2.3.1 λ-versions of simulation methods

We will now briefly discuss 'λ-versions' of the LSTD, LSPE, and TD algorithms, which use a parameter $\lambda \in [0, 1)$,[9] and aim to solve the geometrically weighted multistep version of Bellman's equation $J = T^{(\lambda)} J$, where T is defined by

$$T^{(\lambda)} = (1 - \lambda) \sum_{t=0}^{\infty} \lambda^t T^{t+1}. \tag{30}$$

The best known example is TD(λ) for α-discounted problems [28]. The motivation for using $\lambda > 0$ is that the error $\|J^* - \Phi r^*\|_\infty$ corresponding to the solution r^*_λ of the projected equation

$$\Phi r = \Pi T^{(\lambda)}(\Phi r),$$

[9] It is also possible to let $\lambda = 1$ with appropriate changes in various formulas that follow, but for simplicity we will not consider this possibility here.

tends to decrease as λ increases, and indeed, an improved error bound similar to the one of Proposition 2 c) can be shown for $\lambda > 0$ (see, e.g., Proposition 6.3.5 in [18]). However, the number of samples needed to approximate well the projected equation tends to increase; this is the well-known bias-variance tradeoff, and we refer to textbook sources, such as [1, 2], and [18] for a discussion. In this survey, we mostly focus on $\lambda = 0$, but our algorithms and qualitative conclusions generally apply to $\lambda \in [0, 1)$ (for further discussion, see [24], where extensions of LSTD(λ), LSPE(λ), and TD(λ) beyond the MDP context are also provided). Let us note that the projected equation $\Phi r = \Pi T^{(\lambda)}(\Phi r)$ can be written as a linear equation of the form $C^{(\lambda)} r = d^{(\lambda)}$, where similar to (9), $C^{(\lambda)}$ and $d^{(\lambda)}$ are defined by

$$C^{(\lambda)} = \Phi' \Xi (I - A^{(\lambda)})\Phi, \;\; d^{(\lambda)} = \Phi' \Xi g^{(\lambda)}$$

with

$$A^{(\lambda)} = (1-\lambda) \sum_{t=0}^{\infty} \lambda^t A^{t+1}, \;\; g^{(\lambda)} = \sum_{t=0}^{\infty} \lambda^t A^t g.$$

Similar to the case $\lambda = 0$, there are simulation procedures that produce approximations $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ to $C^{(\lambda)}$ and $d^{(\lambda)}$, respectively, thereby leading to corresponding matrix inversion and iterative methods. However, for these procedures to be algorithmically convenient, some restrictions must be observed. In particular, it is essential that row and column sampling is generated by a common process, so that

$$j_k = i_{k+1}, \;\; k = 0, 1, \ldots,$$

where each transition (i_k, i_{k+1}) is generated using the matrix L (which we recall may be unrelated to the matrix A of the system $J = g + AJ$, other than $\ell_{ij} > 0$ if $a_{ij} > 0$, cf. equation (19)). In this case, approximations $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ to $C^{(\lambda)}$ and $d^{(\lambda)}$, respectively, are obtained from the following recursive algorithm, first given by Bertsekas and Yu [24]: [10]

$$C_k^{(\lambda)} = \Big(1 - \frac{1}{k+1}\Big) C_{k-1}^{(\lambda)} + \frac{1}{k+1} z_k \Big(\phi(i_k) - \frac{a_{i_k i_{k+1}}}{\ell_{i_k i_{k+1}}} \phi(i_{k+1})\Big)', \quad (31)$$

$$d_k^{(\lambda)} = \Big(1 - \frac{1}{k+1}\Big) d_{k-1}^{(\lambda)} + \frac{1}{k+1} z_k g(i_k), \quad (32)$$

where z_k is generated by

$$z_k = \lambda \frac{a_{i_{k-1} i_k}}{\ell_{i_{k-1} i_k}} z_{k-1} + \phi(i_k). \quad (33)$$

An analysis of the convergence $C_k^{(\lambda)} \to C^{(\lambda)}$ and $d_k^{(\lambda)} \to d^{(\lambda)}$ has been given in several sources under different assumptions:

a) In Nedić and Bertsekas [61] for the case of policy evaluation in an α-discounted problem, where A/α is the transition probability matrix P and row and column sampling is done according to that matrix ($Q = L = P$). This is the case where there is no exploration (cf. the subsequent discussion in Section 3.2).

b) In Bertsekas and Yu [24], assuming that $\lambda \max_{(i,j)} \frac{a_{ij}}{\ell_{ij}} < 1$ (where we adopt the convention $0/0 = 0$), in which case the vectors z_k of (33) are generated by a contractive process and remain bounded. This covers the common case in MDP where $L = (1-e)P + e\bar{P}$ where $e > 0$ is a small constant, P is a transition matrix corresponding to the policy under evaluation, and $\bar{P}$ is a transition probability matrix that induces exploration. The essential restriction here is that λ should be no more than $(1 - e)$.

c) In Yu [41, 42], for all $\lambda \in [0, 1]$ and $\lambda \sum_{j=1}^{n} a_{ij} < 1$ for all i, in which case the vectors z_k typically become unbounded as k increases when $\lambda \max_{(i,j)} \frac{a_{ij}}{\ell_{ij}} > 1$. Mathematically, this is the most challenging analysis and involves interesting stochastic phenomena.

Much of what will be said later for projected equation methods where $\lambda = 0$ also applies to methods with $\lambda > 0$.

We finally mention a potential drawback of the row and column sampling methods: they generally require explicit knowledge of the ratios a_{ij}/ℓ_{ij}. Sometimes, in 'model-free' applications of approximate DP, the matrix A is not explicitly known and can only be computed through simulation, so in this case row and column sampling may not apply, at least in the general form given here. We will discuss this issue further in Section 3.8 in the context of Q-learning.

2.4 Simulation by importance sampling

We have seen that a core issue in approximate policy evaluation using the projected equation is to estimate the matrix C by sampling (also to estimate d, but this is similar and simpler). We may thus consider general sampling methods for estimating matrices of the form

$$Z = DH\Phi$$

for an $s \times n$ matrix D, an $n \times n$ matrix H, and an $n \times s$ matrix Φ (in our case, $D = \Phi' \Xi$ and $H = I - A$, or $H = I$ and $H = A$). When n is very large and s is relatively small, we may estimate each of the components of Z separately. Denoting by $d_{\ell i}$, h_{ij}, and ϕ_{jq} the components of D, H, and Φ, respectively, the (ℓ, q)th component of Z is

$$z_{\ell q} = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{\ell i} h_{ij} \phi_{jq}, \quad (34)$$

so it is a sum of a very large number of terms that can be estimated by simulation using Monte Carlo integration techniques aimed at variance reduction, such as importance sampling.

In particular, we generate samples $(i_0, j_0), \ldots, (i_k, j_k)$ according to some sampling distribution $\{\zeta_{ij} | i, j = 1, \ldots, n\}$, and we estimate $z_{\ell q}$ by

$$\hat{z}_{\ell q} = \frac{1}{k+1} \sum_{t=0}^{k} \frac{d_{\ell i_t} h_{i_t j_t} \phi_{j_t q}}{\zeta_{i_t j_t}}.$$

According to the principles of importance sampling (see, e.g., Liu [64] and Rubinstein and Kroese [65]), for fixed ℓ, q, the distribution ζ_{ij} should approximate in form the generic

[10] If $g(i_k)$ is replaced by a simulation sample $g(i_k, i_{k+1})$ the formula for updating $d_k^{(\lambda)}$ should be

$$d_k^{(\lambda)} = \Big(1 - \frac{1}{k+1}\Big) d_{k-1}^{(\lambda)} + \frac{1}{k+1} z_k \frac{a_{i_k i_{k+1}}}{\ell_{i_k i_{k+1}}} g(i_k, i_{k+1}).$$

term $d_{\ell i}h_{ij}\phi_{jq}$ of the sum, i.e.,

$$\zeta_{ij} \approx \frac{d_{\ell i}h_{ij}\phi_{jq}}{N},$$

where N is a normalization constant (we assume here that $d_{\ell i}h_{ij}\phi_{jq}$ is nonnegative for all (i,j), otherwise the sum (34) should be split into the two sums corresponding to the positive and the negative terms).

A key issue here is the design of the distribution ζ_{ij}, which by necessity should exploit the special/sparsity structure of the matrices D, B, and Φ, to localize the distribution in a relatively small portion of the set of pairs (i,j). When a suitable importance sampling distribution can be obtained, the variance of the estimates $\hat{z}_{\ell q}$ and the attendant number of samples for a given accuracy can be dramatically reduced (by several orders of magnitude). We do not pursue this topic further, but refer to the papers [50, 66] for sampling distribution design techniques that have been used successfully in some very large problems within a related least-squares estimation context.

2.5 Convergence rate issues

Let us now discuss the choice of γ and G_k in the iterative method

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k), \tag{35}$$

from the convergence rate point of view. It can be easily verified with simple examples that, in the deterministic case where $G_k \equiv G$, $C_k \equiv C$, and $d_k \equiv d$, the values of γ and G affect significantly the convergence rate of the deterministic iteration (11). This is evident when $I - \gamma GC$ is a contraction, since then the convergence rate is linear and is determined by the spectral radius of $I - \gamma GC$. Surprisingly, however, when G_k, C_k, and d_k are generated by the preceding simulation schemes, the asymptotic convergence rate of the simulation-based iteration (35) does not depend on the choices of γ and G. Indeed, it can be proved that the iteration (35) converges at the same asymptotic rate, for all choices of γ and G, as long as $I - \gamma GC$ is a contraction (although the short-term convergence rate may be significantly affected by the choices of γ and G).

The reason is that the deterministic iteration has a linear convergence rate (since it involves the contraction $I - \gamma GC$), which is fast relative to the slow convergence rate of the simulation-generated G_k, C_k, and d_k. Thus, the simulation-based iteration (35) operates on two time scales (see, e.g., Chapter 6 in [67]): the slow time scale at which G_k, C_k, and d_k change, and the fast time scale at which r_k adapts to changes in G_k, C_k, and d_k. As a result, essentially, there is convergence in the fast time scale before there is appreciable change in the slow time scale. Roughly speaking, r_k 'sees G_k, C_k, and d_k as effectively constant,' so that for large k, r_k is essentially equal to the corresponding limit of iteration (35) with G_k, C_k, and d_k held fixed. When C_k is invertible, this limit is $C_k^{-1}d_k$. It follows that the sequence r_k generated by the scaled LSPE iteration (35) 'tracks' the sequence $C_k^{-1}d_k$ generated by the LSTD iteration in the sense that

$$\|r_k - C_k^{-1}d_k\| \ll \|r_k - r^*\|, \quad \text{for large } k,$$

independent of the choice of γ and the scaling matrix G that is approximated by G_k (see also [39] for further discussion). Results of this type were conjectured for the LSPE method in [62] and were formally proven in [51] for discounted and average cost problems and for the special choice (25) of G_k. The proof for more general cases where C is invertible should be similar but has not yet been documented.

2.6 Regression-based methods for projected equations

We will now consider the case where the matrix C in the projected equation is nearly singular and/or the simulation errors $C - C_k$ and $d - d_k$ are relatively large. Then, the matrix inversion/LSTD method encounters serious difficulties. To get some understanding into this, consider the approximate inversion of a small nonzero number c, which is estimated with simulation error e. The absolute and relative errors are

$$E = \frac{1}{c+e} - \frac{1}{c}, \quad E_r = \frac{E}{1/c}.$$

By a first-order Taylor series expansion around $e = 0$, we obtain for small e

$$E \approx \frac{\partial\left(\frac{1}{c+e}\right)}{\partial e}\Big|_{e=0} e = -\frac{e}{c^2}, \quad E_r \approx -\frac{e}{c}.$$

Thus, for the estimate $1/(c+e)$ to be reliable, we must have $|e| \ll |c|$. If N independent samples are used to estimate c, the variance of e is proportional to $1/N$, so for a small relative error, $E_r \ll 1$, we must have $N \gg 1/c^2$. Thus, as c approaches 0, the amount of sampling required for reliable simulation-based inversion increases very fast. This is similar to the roundoff error-related difficulties of nearly singular matrix inversion.

To counter the large LSTD errors associated with a near-singular matrix C, we may estimate r^* by a form of regularized regression, which works even if the projected equation is singular. In this approach, suggested by Wang et al. [50], instead of solving the system $C_k r = d_k$, we use regression based on an associated linear model that properly encodes the effect of the simulation noise. We will see, however, that this method comes at a price: it introduces a bias of the estimate of r towards some fixed vector/prior guess $\bar{r}$. To reduce or eliminate this bias, it is necessary to adopt an iterative regularization approach: start with some $\bar{r}$, obtain an estimate $\hat{r}_k$, replace $\bar{r}$ by $\hat{r}_k$, and repeat for any number of times. This gives rise to an iterative method, which turns out to be a special case of the method (35).

Given simulation-based estimates C_k and d_k to C and d, respectively, let us write the projected form of Bellman's equation $d = Cr$ as

$$d_k = C_k r + e_k, \tag{36}$$

where e_k is the vector

$$e_k = (C - C_k)r + d_k - d,$$

which we view as 'simulation noise'. We then estimate the solution $r^* = C^{-1}d$ based on (36) by using regression. In particular, we choose r by solving the least-squares problem:

$$\min_r \left[(d_k - C_k r)'\Sigma^{-1}(d_k - C_k r) + \beta\|r - \bar{r}\|^2\right], \tag{37}$$

where $\bar{r}$ is an a priori estimate of r^*, Σ is some positive definite symmetric matrix, and β is a positive scalar. By setting to 0 the gradient of the least-squares objective in (37), we

can find the solution in closed form:

$$\hat{r}_k = (C_k' \Sigma^{-1} C_k + \beta I)^{-1} (C_k' \Sigma^{-1} d_k + \beta \bar{r}). \quad (38)$$

A suitable choice of $\bar{r}$ may be some heuristic guess based on intuition about the problem, or it may be the parameter vector corresponding to the estimated cost vector $\Phi\bar{r}$ of a similar policy (for example, a preceding policy in an approximate policy iteration context). One may try to choose Σ in special ways to enhance the quality of the estimate of r^*, but we will not consider this issue here, and the subsequent analysis in this section does not depend on the choice of Σ, as long as it is positive definite and symmetric.

The quadratic $\beta\|r - \bar{r}\|^2$ in (37) is a regularization term and has the effect of 'biasing' the estimate $\hat{r}_k$ towards the a priori guess $\bar{r}$. The proper size of β is not clear (a large size reduces the effect of near singularity of C_k, and the effect of the simulation errors $C_k - C$ and $d_k - d$, but may cause a large 'bias'). However, this is typically not a major difficulty in practice, because trial-and-error experimentation with different values of β involves low-dimensional linear algebra calculations once C_k and d_k become available.

To eliminate the bias of $\hat{r}_k$ towards $\bar{r}$, one possibility is to adopt an iterative regularization approach: start with some $\bar{r}$, obtain $\hat{r}_k$, replace $\bar{r}$ by $\hat{r}_k$, and repeat for any number of times. This turns LSTD to the iterative method,

$$r_{k+1} = (C_k' \Sigma_k^{-1} C_k + \beta I)^{-1} (C_k' \Sigma_k^{-1} d_k + \beta r_k),$$

where Σ_k are positive definite symmetric matrices that converge to a positive definite matrix Σ, and β is a positive scalar (38). Equivalently, this method can be written as

$$r_{k+1} = r_k - G_k(C_k r_k - d_k), \quad (39)$$

where

$$G_k = (C_k' \Sigma_k^{-1} C_k + \beta I)^{-1} C_k' \Sigma_k^{-1}, \quad (40)$$

and can be recognized as a simulation-based version of the special case of the class of LSPE-type methods (35), whose convergence properties have been discussed in Proposition 5.

The salient feature of the method (39), (40) is that, based on Proposition 5 and experimental evidence, it can be expected to converge to a solution of the projected equation $Cr = d$ even if the mapping ΠT is not a contraction, as long as C is invertible. We will see in the next section that this generality is particularly important in the context of policy iteration where the simulation is done by using exploration-enhanced Markov chains. Furthermore, the iteration is well suited for problems where C is nearly singular.

3 Policy iteration issues

We will now discuss various issues that arise within a policy iteration context: the effects of inadequate exploration, of inexact policy evaluation, and of policy oscillation. For concreteness, we focus on the α-discounted MDP given in the introduction, where the cost vector J_μ of a policy μ is the unique fixed point of the mapping T_μ of (1), and the optimal cost vector J^* is the unique fixed point of the mapping T of (2).

We consider a form of policy iteration where we compute simulation-based approximations Φr_μ to the cost vectors J_μ of stationary policies μ with some policy evaluation method. We use Φr_μ to compute new policies based on (approximate) policy improvement using the formula $T_{\bar{\mu}}(\Phi r_\mu) = T(\Phi r_\mu)$, i.e.,

$$\bar{\mu}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha\phi(j)' r_\mu), \quad i = 1, \ldots, n, \quad (41)$$

where $\phi(j)'$ is the row of Φ that corresponds to state j. The method terminates with μ if $T_\mu(\Phi r_\mu) = T(\Phi r_\mu)$. We then repeat with μ replaced by $\bar{\mu}$.

The theoretical basis for the preceding method was given in Proposition 6.2 in [1], where it was shown that if the policy evaluation is accurate to within δ (in the sup-norm sense $\|\Phi r_\mu - J_\mu\|_\infty \leqslant \delta$), then the method will yield a sequence of policies $\{\mu^k\}$ such that

$$\limsup_{k\to\infty} \|J_{\mu^k} - J^*\|_\infty \leqslant \frac{2\alpha\delta}{(1-\alpha)^2}. \quad (42)$$

An alternative bound based on a Euclidean norm is given by Munos [68]. Experimental evidence indicates that the bound (42), while tight in theory (Example 6.4 in [1]), is usually conservative in practice. Furthermore, often just a few policy evaluations are needed before the bound is attained. Some insight about the looseness of the bound (42) may be obtained by noting that we may add any multiple of the unit vector to Φr_μ, thereby making δ arbitrarily large, without affecting the policy improvement process and hence also the generated sequence $\{\mu^k\}$.

When the policy sequence $\{\mu^k\}$ terminates with some $\bar{\mu}$, the much sharper bound

$$\|J_{\bar{\mu}} - J^*\|_\infty \leqslant \frac{2\alpha\delta}{1-\alpha} \quad (43)$$

holds. To show this, let $\bar{J}$ be the cost vector $\Phi r_{\bar{\mu}}$ obtained by policy evaluation of $\bar{\mu}$, and note that it satisfies $\|\bar{J} - J_{\bar{\mu}}\|_\infty \leqslant \delta$ (by our assumption on the accuracy of policy evaluation) and $T\bar{J} = T_{\bar{\mu}}\bar{J}$ (since μ^k terminates at $\bar{\mu}$). We write

$$\begin{aligned} TJ_{\bar{\mu}} &\geqslant T(\bar{J} - \delta e) = T\bar{J} - \alpha\delta e = T_{\bar{\mu}}\bar{J} - \alpha\delta e \\ &\geqslant T_{\bar{\mu}}(J_{\bar{\mu}} - \delta e) - \alpha\delta e = T_{\bar{\mu}}J_{\bar{\mu}} - 2\alpha\delta e \\ &= J_{\bar{\mu}} - 2\alpha\delta e, \end{aligned}$$

where e is the unit vector, from which by applying T to both sides, we obtain

$$T^2 J_{\bar{\mu}} \geqslant TJ_{\bar{\mu}} - 2\alpha^2\delta e \geqslant J_{\bar{\mu}} - 2\alpha\delta(1+\alpha)e,$$

and by similar continued application of T to both sides,

$$J^* = \lim_{m\to\infty} T^m J_{\bar{\mu}} \geqslant J_{\bar{\mu}} - \frac{2\alpha\delta}{1-\alpha} e,$$

thereby showing the error bound (43).

An important fact, to be shown later, is that when the aggregation approach is used for policy evaluation, the policy sequence $\{\mu^k\}$ terminates with some $\bar{\mu}$, while this is generally not true for the projected equation approach.

3.1 The issue of policy computations

We first note briefly an important practical issue: the computation of the improved policy via the minimization of (41) may present a serious difficulty when the expected value in (41) is hard to compute, and/or when the number of controls is large (or infinite in an extended version of the method where the control constraint set is infinite). Moreover, this issue arises generically when an approximate cost function

is translated to an online implementation of a corresponding policy.

A number of devices have been proposed to address this difficulty, such as various forms of Q-learning, model simplifications (such as post-decision states), and various forms of approximate minimization in (41) (such as adaptive sampling). We refer to the literature, and particularly the books [5, 7, 9] for an explanation of these terms and further discussion. Note that in some cases, the choice of the approximation architecture may be specially designed to exploit special problem structure and facilitate the minimization.

3.2 The issue of exploration

We next discuss an important generic difficulty with simulation-based policy iteration: to evaluate a policy μ, we need to generate cost samples using that policy, but this biases the simulation by underrepresenting states that are unlikely to occur under μ. As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate, causing potentially serious errors in the calculation of the improved control policy $\bar{\mu}$ via the policy improvement (41). A related issue is the presence of transient states in the Markov chain used for simulation (in which case the components of ξ corresponding to transient states are 0), or multiple recurrent classes (in which case some states may never be generated during the simulation).

The difficulty just described is known as inadequate exploration of the system's dynamics because of the use of a fixed policy. It is a particularly acute difficulty when the system is deterministic, or more generally when the randomness embodied in the transition probabilities is 'relatively small'. One possibility for guaranteeing adequate exploration of the state space is to frequently restart the simulation and to ensure that the initial states employed form a rich and representative subset. There are also other related possibilities. For example, an approach called iterative resampling, is to enrich the sampled set of states in evaluating the current policy μ as follows: derive an initial cost evaluation of μ, simulate the next policy $\bar{\mu}$ obtained on the basis of this initial evaluation to obtain a set of representative states $\bar{S}$ visited by $\bar{\mu}$, and repeat the evaluation of μ using additional trajectories initiated from $\bar{S}$.

One of the most frequently used approaches is to artificially introduce some extra randomization in the simulation, by occasionally modifying the state sequence generated by the policy μ. In a common scheme of this type, we do row sampling according to an irreducible transition probability matrix

$$Q = (I - B)P + B\bar{P}, \tag{44}$$

where P is the transition matrix corresponding to μ, B is a diagonal matrix with diagonal components $\beta_i \in [0, 1]$, and $\bar{P}$ is another transition matrix. Thus, at state i, the next state is generated with probability $1 - \beta_i$ according to transition probabilities p_{ij} corresponding to μ, and with probability β_i according to transition probabilities $\bar{p}_{ij}$ whose purpose is to enhance exploration. [11]

Column sampling can be done using $L = P$, in which case the policy evaluation phase finds a solution to the projected equation $\Phi r = \Pi T(\Phi r)$, but with an important difference: the projection is with respect to the norm $\|\cdot\|_\xi$ where ξ is the steady-state distribution of Q rather than P. As a result of this norm mismatch, ΠT may not be a contraction, and a policy evaluation method that does not rely on contractions is needed. [12] LSTD is one such method, and the regression-based scaled LSPE method (39), (40) is another.

There is a further restriction when a λ-method is used in conjunction with simulation. As discussed in Section 2.3, one must then choose the same Markov chain for row and column sampling ($Q = L$) in order to use the algorithm (31)–(33) for computing simulation-based $C_k^{(\lambda)} \approx C^{(\lambda)}$ and $d_k^{(\lambda)} \approx d^{(\lambda)}$. Still in this case, both LSTD and the regression-based scaled LSPE method (39), (40) may be used, even if $\Pi T^{(\lambda)}$ is not known to be a contraction (we refer to [24] and [70, 71] for an analysis and discussion). It is then necessary, however, to know explicitly the ratios $a_{ij}/\ell_{ij} = \alpha p_{ij}/q_{ij}$ that appear in the algorithm (31)–(33). In practical applications reported in the literature, this difficulty has been bypassed by disregarding the ratios p_{ij}/q_{ij} from the formulas (31)–(33) (i.e., replacing a_{ij}/ℓ_{ij} with just α).

We mention that there are also some other possibilities for exploration enhancement for which we refer to [24] and [9]. Moreover, the TD(0) method has been modified to address the exploration-related convergence difficulties (Sutton et al. [72], Maei et. al. [73], Sutton et. al. [74]). We finally note that exploration becomes an even more acute difficulty in the context of policy iteration for computing Q-factors, as we will discuss in Section 3.8.

3.3 Limited sampling/optimistic versions

In the LSTD and LSPE methods discussed so far, the underlying assumption is that each policy is evaluated with a very large number of samples, so that accurate approximations of C and d are obtained. There are also optimistic versions, where the policy μ is replaced by an 'improved' policy $\bar{\mu}$ after only a limited number of simulation samples have been processed. The motivation for such versions is potentially faster progress towards better policies and less susceptibility to exploration difficulties (one may argue that mixing samples from several past policies may have a beneficial exploration effect). While experimental evidence lends some support to this speculation, solid evidence for the validity of optimistic methods is currently lacking, at

[11] In the literature, e.g., [2], the policy being evaluated is sometimes called the target policy to distinguish from the matrix $\bar{P}$ that may correspond to a policy used for exploration, which is called behavior policy. Also, methods that use a behavior policy are called off-policy methods, while methods that do not are called on-policy methods. Note, however, that $\bar{P}$ need not correspond to an admissible policy, and indeed, there may not exist a suitable admissible policy that can induce sufficient exploration. For example, in many queueing applications, the transition matrices corresponding to all policies (including randomized ones) are very sparse and cannot induce adequate exploration.

[12] Proposition 2 a) applies only for small values of the probabilities β_i along the diagonal of B. It also applies only for λ sufficiently close to 1, in the case of the λ-methods of Section 2.3. The reason is that the operator $T^{(\lambda)}$ of (30) (and hence also $\Pi T^{(\lambda)}$) can be shown to be a contraction of modulus arbitrarily close to 0 if λ is sufficiently close to 1 (see, e.g., [18], Proposition 6.3.3, or [9, 69]).

least for the case of projected equation-based policy evaluation, when the chattering phenomenon may occur, as will be explained shortly. The validity of optimistic policy iteration is better established in the case of a lookup table representation (where $\Phi = I$; see Section 5.4 in [1]) and the related case of aggregation-based policy evaluation (see Section 4). However, even in that case, the convergence properties of optimistic policy iteration are fragile and depend on the implementation details, as discussed in Section 5.4 in [1]; see also Williams and Baird [19] who provide remarkable counterexamples to the convergence of a version of asynchronous optimistic policy iteration where costs and policies are updated one at a time (these examples do not involve cost function approximation or simulation).

A natural form of optimistic LSTD is $\hat{r}_{k+1} = C_k^{-1} d_k$, where C_k and d_k are obtained by averaging samples collected using the controls corresponding to the (approximately) improved policy. By this we mean that C_k and d_k are time averages of the matrices and vectors

$$\phi(i_t)\big(\phi(i_t) - \alpha\phi(i_{t+1})\big)', \;\; \phi(i_t)g(i_t, i_{t+1}),$$

corresponding to simulated transitions (i_t, i_{t+1}) that are generated using the policy μ^{k+1} whose controls are given by

$$\mu^{k+1}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha\phi(j)'\hat{r}_k).$$

Unfortunately, this method requires the collection of many samples between policy updates, as it is susceptible to simulation noise in C_k and d_k, particularly when C_k is nearly singular. The optimistic version of the (scaled) LSPE method

$$r_{k+1} = r_k - \gamma G_k(C_k r_k - d_k)$$

is based on similar ideas, but seems less susceptile to simulation noise, because of its iterative nature. Note that because the theoretical convergence guarantees of LSPE apply only to the nonoptimistic version, it may be essential to experiment with various values of the stepsize γ.

To improve the reliability of the optimistic LSTD method, it seems necessary to turn it into an iterative method, which then brings it very close to LSPE. In particular, an iterative version of the regression-based LSTD method (38) is given by (39) and is the special case of LSPE, corresponding to the special choice of the scaling matrix G_k of (40). This method, as well as other related versions of LSPE, is less susceptible to the high level of simulation noise associated with limited number of samples between policy updates in optimistic policy iteration.

3.4 Constrained policy iteration

It is natural in approximate DP to try to exploit whatever prior information is available about J^*, such as for example bounds on $J^*(i)$ for some or all of the states i. If it is known that J^* belongs to a subset of $\mathbb{R}^n$, we may try to find an approximation Φr that belongs to that subset. This leads to projected equations of the form $\Phi r = \Pi T(\Phi r)$, where Π is the operator that projects a vector onto a closed convex subset $\hat{S}$ of the subspace $S = \{\Phi r | r \in \mathbb{R}^s\}$, a form of nonlinear approximation. Corresponding analogs of LSTD- and LSPE-type methods for such projected equations involve the solution of linear variational inequalities rather than linear systems of equations and are described in [39]. In particular, it is shown in [39] that $\hat{r}$ solves the equation $\Phi\hat{r} = \Pi T(\Phi\hat{R})$ if and only if it solves the variational inequality

$$(C\hat{r} - d)'(r - \hat{r}) \geqslant 0, \;\; \forall r \in \hat{R}, \tag{45}$$

where $\hat{R} = \{r | \Phi r \in \hat{S}\}$ (cf. (9)). This is a generalization of the equation $Cr = d$, which corresponds to the case $\hat{R} = \mathbb{R}^s$. Thus, one may carry out simulation-based policy evaluation by replacing C and d with simulation-based estimates in the preceding variational inequality. A potential difficulty here is that even if the set $\hat{S}$ is simple, the set $\hat{R}$ may be complicated (e.g., it may be specified by a large number of inequalities), in which case one may consider approximating $\hat{R}$ with a more convenient set.

There is a simpler alternative to solving the variational inequality (45) in the practically common case where upper and/or lower bounds of J^* are available. In this alternative, which stems from the work of Bertsekas and Yu [75], we introduce the bounds directly into the approximation architecture and use a corresponding nonlinear iterative policy evaluation algorithm. As an example, suppose that we have access to the components of a vector $\bar{J}$ with $\bar{J}(i) \geqslant J^*(i)$ for all i (for example, the cost vector of a known policy). Then, policy iteration with cost function approximation can be modified to incorporate this knowledge as follows. Given a policy μ, we evaluate it by finding an approximation Φr_μ to the solution $\tilde{J}_\mu$ of the equation

$$\tilde{J}_\mu(i) = \sum_{j=1}^{n} p_{ij}(\mu(i))(g(i,\mu(i),j) + \alpha\min\{\bar{J}(j), \tilde{J}_\mu(j)\}), \;\; i = 1, \ldots, n \tag{46}$$

followed by the policy improvement operation (cf. equation (41))

$$\bar{\mu}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha\min\{\bar{J}(j), \phi(j)'r_\mu\}), \;\; i = 1, \ldots, n, \tag{47}$$

where $\phi(j)'$ is the row of Φ that corresponds to state j.

Note that (46) has a unique solution since its right-hand side represents a sup-norm contraction mapping. Indeed, this is Bellman's equation for the Q-factor of an optimal stopping problem involving a stopping cost $\bar{J}(i)$ at state i, a problem that has received a lot of attention in the approximate DP literature because of its favorable structure (see Section 6.8 in [1], or Section 6.4.2 in [18]). In particular, (46) can be solved approximately, with approximation of $\tilde{J}_\mu$ by Φr_μ, using variants of the TD(0) and LSPE(0) methods (see Tsitsiklis and Van Roy [76], Choi and Van Roy [63], and Yu and Bertsekas [77]). One form of the LSPE(0) method of [77] generates a long trajectory $(i_0, i_1, \ldots, i_m)$ using the policy $\bar{\mu}$ of (47), and determines the weight vector $r_{\bar{\mu}}$ as the limit of the iteration

$$r_{k+1} = \arg\min_{r\in\mathbb{R}^s} \sum_{t=0}^{m-1} \Big(\phi(i_t)'r - g(i_t, \mu(i_t), i_{t+1}) - \alpha\min\big\{\bar{J}(i_{t+1}), \phi(i_{t+1})'r_k\big\}\Big)^2, \;\; r_0 = r_\mu, \tag{48}$$

i.e., $r_{\bar{\mu}} = \lim_{k\to\infty} r_k$ (for this it is necessary that the length of the simulation trajectory is sufficiently large). This is the

analog of the batch simulation approach of (14): first a batch of simulation data is generated, and then the corresponding Bellman equation (46) is solved approximately via the iteration (48) (an LSTD/matrix inversion approach is not possible because (46) is nonlinear). The method (48) involves substantial overhead because it involves the entire simulation trajectory at each iteration. There are also alternative methods given by [77], which are more iterative in character (iterate on r as new simulation data is generated) and also require less overhead per iteration; see also [9, 18] for details. [13]

Under the assumption $\bar{J}(i) \geqslant J^*(i)$ for all i, and a lookup table representation ($\Phi = I$), it can be shown that the method (46), (47) yields J^* in a finite number of iterations, just like the standard (exact) policy iteration method. The proof is based on the monotonicity and sup-norm contraction properties of the mappings $\bar{T}_\mu$ and $\bar{T}$ given by

$$(\bar{T}_\mu J)(i) = \sum_{j=1}^{n} p_{ij}(\mu(i))(g(i,\mu(i),j) + \alpha \min\{\bar{J}(j), J(j)\}), \quad i = 1, \ldots, n, \quad (49)$$

$$(\bar{T}J)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha \min\{\bar{J}(j), J(j)\}), \quad i = 1, \ldots, n, \quad (50)$$

(46) and (47), and the fact that J^* is the unique fixed point of F (which rests on the hypothesis $\bar{J} \geqslant J^*$); see the subsequent Proposition 6.

When a compact feature-based representation is used ($\Phi \neq I$), the constrained policy iteration method (46), (47) is still vulnerable to oscillation regardless of which variant of (48) is used (see the next section). The magnitude of the oscillations can be estimated with an error bound similar to (42). However, the method may produce substantially different results than its unconstrained counterpart, because the policies involved in an oscillation may be different in the two methods.

3.5 Policy oscillation and chattering

Projected equation-based variants of policy iteration methods are popular in practice and have been tested extensively, dating to the early nineties (see, e.g., the books [1,2], and the references quoted there; for a sample of more recent experimental studies, see Lagoudakis and Parr [78], Jung and Polani [79], Busoniu et al. [80], and Thiery and Scherrer [49]). However, the associated convergence behavior is complex and involves potentially damaging oscillations, which despite their initial description long ago [81] are still not well understood at present.

To get a sense of this behavior, we introduce the so called greedy partition, discussed in Section 6.4 of [1]. This is a partition of the space $\mathbb{R}^s$ of parameter vectors r into subsets R_μ, each subset corresponding to a stationary policy μ, and defined by

$$R_\mu = \{r | T_\mu(\Phi r) = T(\Phi r)\}, \quad (51)$$

or equivalently

$$R_\mu = \Big\{r \Big| \mu(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)(g(i,u,j) + \alpha\phi(j)'r), \quad i = 1, \ldots, n\Big\}.$$

Thus, R_μ is the set of parameter vectors r for which μ is 'greedy' with respect to Φr. Note that the greedy partition depends only on Φ and is independent of the particular method used for policy evaluation (i.e., whether a projection/TD or different method is used).

For simplicity, let us assume that we use a policy evaluation method that for each given μ produces a unique parameter vector r_μ. Nonoptimistic policy iteration starts with a parameter vector r_0, which specifies μ^0 as a greedy policy with respect to Φr_0 and generates r_{μ^0} by using the given policy evaluation method. It then finds a policy μ^1 that is greedy with respect to Φr_{μ^0}, i.e., a μ^1 such that $r_{\mu^0} \in R_{\mu^1}$. It then repeats the process with μ^1 replacing μ^0. If some policy μ^k satisfying $r_{\mu^k} \in R_{\mu^k}$ is encountered, the method keeps generating that policy. This is the necessary and sufficient condition for policy convergence in the nonoptimistic policy iteration method. Of course, the mere fact that a policy iteration method is guaranteed to converge is not in itself a guarantee of good performance when cost function approximation is used, beyond the fact that a better error bound holds in this case (43) versus (42).

In the case of a lookup table representation where the parameter vectors r_μ are equal to the cost-to-go vector J_μ, the condition $r_{\mu^k} \in R_{\mu^k}$ is equivalent to $r_{\mu^k} = Tr_{\mu^k}$ and is satisfied if and only if μ^k is optimal. When there is function approximation, however, this condition need not be satisfied for any policy. Since there is a finite number of possible vectors r_μ, one generated from another in a deterministic way, the algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}}, r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, r_{\mu^{k+m-1}} \in R_{\mu^{k+m}}, \quad r_{\mu^{k+m}} \in R_{\mu^k}, \quad (52)$$

(see Fig. 2). Furthermore, there may be several different cycles, and the method may end up converging to any one of them. The actual cycle obtained depends on the initial policy μ^0. This is similar to gradient methods applied to minimization of functions with multiple local minima, where the limit of convergence depends on the starting point.

In the case of optimistic policy iteration, the trajectory of the method is less predictable and depends on the fine details of the iterative policy evaluation method, such as the frequency of the policy updates and the stepsize used. Generally, given the current policy μ, optimistic policy iteration will move towards the corresponding 'target' parameter r_μ, for as long as μ continues to be greedy with respect to the current cost-to-go approximation Φr, that is, for as long as the current parameter vector r belongs to the set R_μ. Once, however, the parameter r crosses into another set, say $R_{\bar{\mu}}$,

[13] An algorithm similar to (46)–(48) is also possible if we know instead a lower bounding vector to J^* or both upper and lower bounding vectors: the terms $\min\{\bar{J}(j), \tilde{J}_\mu(j)\}$ and $\min\{\bar{J}(j), \phi(j)'r_\mu\}$ in (46) and (47) should be replaced by the respective projections of $\tilde{J}_\mu(j)$ or $\phi(j)'r_\mu$ on the corresponding interval of upper and/or lower bounds. In this case, a generalization of the optimal stopping algorithm of [77], given in Section 7.3 of [24], may be used for policy evaluation. Note that this algorithm provides a means for approximating the cost $J_\mu(j)$ of a policy μ with a simple nonlinear architecture $f_j(\phi(j)'r)$, where f_j denotes projection onto the interval of upper and lower bounds to $J^*(j)$. This approach may make sense regardless of whether true upper and/or lower bounds of $J^*(j)$ are known.

the policy $\bar{\mu}$ becomes greedy, and r changes course and starts moving towards the new 'target' $r_{\bar{\mu}}$. Thus, the 'targets' r_μ of the method and the corresponding policies μ and sets R_μ may keep changing, similar to nonoptimistic policy iteration. Simultaneously, the parameter vector r will move near the boundaries that separate the regions R_μ that the method visits, following reduced versions of the cycles that nonoptimistic policy iteration may follow. Furthermore, as Fig. 2 suggests, if diminishing parameter changes are made between policy updates (such as, for example, when a diminishing stepsize is used by the policy evaluation method) and the method eventually cycles between several policies, the parameter vectors will tend to converge to the common boundary of the regions R_μ corresponding to these policies. This is the so-called chattering phenomenon for optimistic policy iteration, whereby there is simultaneously oscillation in policy space and convergence in parameter space. The following is a simple example of policy oscillations and chattering. Other examples are given in Section 6.4.2 in [1] (Examples 6.9 and 6.10).

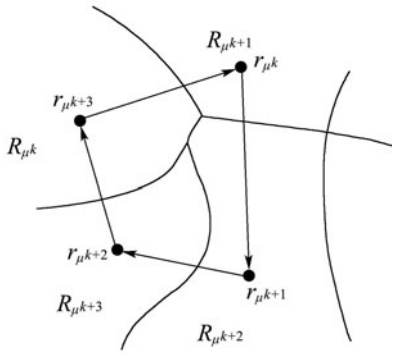

Fig. 2 Greedy partition and cycle of policies generated by nonoptimistic policy iteration. In particular, μ yields $\bar{\mu}$ by policy improvement if and only if $r_\mu \in R_{\bar{\mu}}$. In this figure, the method cycles between four policies and the corresponding four parameters r_{μ^k}, $r_{\mu^{k+1}}$, $r_{\mu^{k+2}}$, and $r_{\mu^{k+3}}$.

Example 1 (Policy oscillation and chattering) Consider a discounted problem with two states, 1 and 2, illustrated in Fig. 3 (a). There is a choice of control only at state 1, and there are two policies, denoted μ^* and μ. The optimal policy μ^*, when at state 1, stays at 1 with probability $p > 0$ and incurs a cost $c < 0$ while, when at state 2, returns to state 1 with 0 cost. The other policy is μ and cycles between the two states with 0 cost. We consider linear approximation with a single feature $\phi(i)' = i$ for each of the states $i = 1, 2$, i.e.,

$$\Phi = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \tilde{J} = \Phi r = \begin{bmatrix} r \\ 2r \end{bmatrix}.$$

We first derive the greedy partition using (51):

$$\begin{aligned} R_{\mu^*} &= \big\{r | p\big(c + \alpha(1 \cdot r)\big) + (1-p)\alpha(2 \cdot r) \leqslant \alpha(2 \cdot r)\big\} \\ &= \{r | c \leqslant \alpha r\}, \\ R_\mu &= \{r | c \geqslant \alpha r\}. \end{aligned}$$

We next calculate the points r_μ and r_{μ^*} that solve the projected equations

$$C_\mu r_\mu = d_\mu, \quad C_{\mu^*} r_{\mu^*} = d_{\mu^*},$$

which correspond to μ and μ^*, respectively. We have

$$\begin{aligned} C_\mu &= \Phi' \Xi_\mu (1 - \alpha P_\mu)\Phi = [1 \;\; 2] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\alpha \\ -a & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\ &= 5 - 9\alpha, \\ d_\mu &= \Phi' \Xi_\mu g_\mu = [1 \;\; 2] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, \end{aligned}$$

so

$$r_\mu = 0.$$

Similarly, with some calculation,

$$\begin{aligned} C_{\mu^*} &= \Phi' \Xi_{\mu^*} (1 - \alpha P_{\mu^*})\Phi \\ &= [1 \;\; 2] \begin{bmatrix} \dfrac{1}{2-p} & 0 \\ 0 & \dfrac{1-p}{2-p} \end{bmatrix} \begin{bmatrix} 1 - \alpha p & -\alpha(1-p) \\ -a & 1 \end{bmatrix} [1 \;\; 2] \\ &= \frac{5 - 4p - \alpha(4 - 3p)}{2 - p}, \\ d_{\mu^*} &= \Phi' \Xi_{\mu^*} g_{\mu^*} = [1 \;\; 2] \begin{bmatrix} \dfrac{1}{2-p} & 0 \\ 0 & \dfrac{1-p}{2-p} \end{bmatrix} \begin{bmatrix} c \\ 0 \end{bmatrix} = \frac{c}{2-p}, \end{aligned}$$

so

$$r_{\mu^*} = \frac{c}{5 - 4p - \alpha(4 - 3p)}.$$

We now note that since $c < 0$, $r_\mu = 0 \in R_{\mu^*}$, while for $p \approx 1$ and $\alpha > 1 - \alpha$, we have $r_{\mu^*} \approx c/(1-\alpha) \in R_\mu$; cf. Fig. 3 (b). In this case, policy iteration cycles between μ and μ^*. Its optimistic version uses some algorithm that moves the current value r towards r_{μ^*} if $r \in R_{\mu^*}$, and towards r_μ if $r \in R_\mu$. Thus, optimistic policy iteration starting from a point in $\mathbb{R}_\mu$ moves towards r_{μ^*} and once it crosses the boundary point c/α of the greedy partition, it reverses course and moves towards r_μ. If the method makes small incremental changes in r before checking whether to change the current policy, it will incur a small oscillation around c/α. If the incremental changes in r are diminishing, the method will converge to c/α. Yet, c/α does not correspond to any one of the two policies and has no meaning as a desirable parameter value.

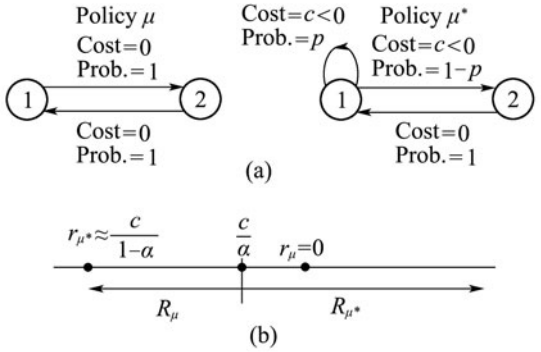

Fig. 3 The problem of Example 1. (a) Costs and transition probabilities for the policies μ and μ^*. (b) The greedy partition and the solutions of the projected equations corresponding to μ and μ^*. Nonoptimistic policy iteration oscillates between r_μ and r_{μ^*}.

Notice that it is hard to predict when an oscillation will occur and what kind of oscillation it will be. For example, if $c > 0$, we have $r_\mu = 0 \in R_\mu$, while for $p \approx 1$ and

$\alpha > 1 - \alpha$, we have $r_{\mu^*} \approx c/(1-\alpha) \in R_{\mu^*}$. Then, approximate and optimistic policy iteration converge to μ (or μ^*) if started with r in R_μ (or R_{μ^*}, respectively).

When chattering occurs, the limit of optimistic policy iteration tends to be on a common boundary of several subsets of the greedy partition and may not meaningfully represent a cost approximation of any of the corresponding policies, as illustrated by the preceding example. Thus, the limit to which the method converges cannot always be used to construct an approximation of the cost-to-go of any policy or the optimal cost-to-go. As a result, at the end of optimistic policy iteration and in contrast with the nonoptimistic version, one must go back and perform a screening process; that is, evaluate by simulation the many policies generated by the method starting from the initial conditions of interest and select the most promising one. This is a disadvantage of optimistic policy iteration that may nullify whatever practical rate of convergence advantages the method may have over its nonoptimistic counterpart.

An additional insight is that the choice of the iterative policy evaluation method (e.g., LSTD, LSPE, or TD for various values of λ) makes a difference in rate of convergence, but does not seem crucial for the quality of the final policy obtained (as long as the methods converge). The choice of method affects the targets r_μ somewhat, but leaves the greedy partition unchanged. As a result, different methods 'fish in the same waters' and tend to yield similar ultimate cycles of policies. Generally, there is no guarantee that any of the policies involved in oscillation is particularly good; the policy iteration process may be just cycling in a 'bad' part of the greedy partition. Moreover, it is hard to know how far from the optimum the oscillation is occurring, similar to the case of local minima in gradient-based optimization.

The full ramifications of policy oscillation in practice are not fully understood at present, but it is clear that they give serious reason for concern. An interesting case in point is the game of tetris, which has been used as a testbed for approximate DP methods [36,46,82–86]. [14] Using policy iteration with policy evaluation based on the projected equation and a set of 22 features, an average score of a few thousands was achieved (originally, the LSPE method was used [36]; similar results were obtained with the LSTD method [78]). Using the same features and a random search method (the cross-entropy method [65, 87]) in the space of weight vectors r, an average score of 600,000 to 900,000 was achieved [85, 88]. The causes of this notable failure of approximate policy iteration have not been clarified yet. It is an open question whether it is due to the inherent difficulty of the tetris problem or whether it can be attributed to inherent pathologies of approximate policy iteration, such as oscillations/chattering between relatively poor policies, or weaknesses of the cost function approximation philosophy. [15]

3.6 Conditions for policy convergence

The preceding analysis has indicated that it is desirable to avoid policy oscillation in policy iteration methods. Moreover, as mentioned earlier, when policies converge, there is a more favorable error bound associated with the method (43) versus (42). It is therefore interesting to investigate conditions under which we have convergence of policies. From the mathematical point of view, it turns out that policy oscillation is caused by the lack of monotonicity and the dependence (through Ξ) on μ of the projection operator. With this in mind, we will replace Π with a constant operator W that has a monotonicity property. We develop convergence conditions for an abstract approximate policy iteration algorithm that applied to general (beyond discounted) DP problems. This leads to conceptual and notational simplification as well as a more broadly applicable algorithm and analysis.

To this end, consider any (possibly nonlinear) mapping $H_\mu : \mathbb{R}^n \mapsto \mathbb{R}^n$, parametrized by the policy μ, and the mapping $H : \mathbb{R}^n \mapsto \mathbb{R}^n$, defined by

$$HJ = \min_{\mu \in \mathcal{M}} H_\mu J, \tag{53}$$

where $\mathcal{M}$ is a finite subset of policies, and the minimization above is done separately for each component of $H_\mu J$; i.e.,

$$(HJ)(i) = \min_{\mu \in \mathcal{M}} (H_\mu J)(i), \quad \forall i = 1, \ldots, n.$$

Abstract mappings of this type and their relation to DP have been studied in Denardo [94], Bertsekas [95], and Bertsekas and Shreve [96]. A special case is the discounted DP case, where $H_\mu = T_\mu$ and $H = T$. Another special case is a mapping H_μ that is similar to T_μ but arises in discounted semi-Markov problems. A special case of a nonlinear mapping H_μ arises in constrained policy iteration, cf. the mapping $\bar{T}_\mu$ of (49). Nonlinear mappings H_μ also arise in the context of minimax DP problems and sequential games; see Shapley [97], and [94–96].

We will construct a policy iteration method that aims to find an approximation to a fixed point of H and evaluates a policy $\mu \in \mathcal{M}$ with a solution $\tilde{J}_\mu$ of the following fixed

[14] Local attraction-type phenomena may be causing similar difficulties in other related approximate DP methodologies: policy iteration with function approximation using the Bellman error method, policy gradient methods, and approximate linear programming (the tetris problem, using the same 22 features, has been addressed by approximate linear programming [82, 83], and with a policy gradient method [84], also with an achieved average score of a few thousands, roughly comparable to the ones obtained using policy iteration). These connections are interesting subjects for investigation.

[15] For a given policy μ, the policy evaluation methods based on the projected equation estimate the cost function of μ by Φr_μ, yet the next policy $\bar{\mu}$ obtained by policy improvement is determined by the Q-factor differences $\tilde{Q}_\mu(i,u) - \tilde{Q}_\mu\big(i,\mu(i)\big)$, where

$$\tilde{Q}_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha\phi(j)' r_\mu\big), \quad i = 1, \ldots, n, \quad u \in U(i),$$

is the corresponding approximation of the Q-factor $Q_\mu(i,u)$ of μ (cf. 41). Thus, adding a constant to all components of Φr_μ does not affect $\bar{\mu}$. This has led to proposals of policy evaluation methods that aim to estimate directly Q-factor differences or cost function differences (see Werbos [89, 90], Baird et al. [91, 92] and Bertsekas [93]). There is no solid theory regarding the merits of these methods over methods based on cost or Q-factor approximation, but it is clear that there are circumstances where they may offer an advantage (see the discussion of Sections 6.6.2 and 6.11 in [1]). The problem is that policy evaluation based on least-squares criteria tends to capture the large-scale behavior of the function being approximated and may thus be less sensitive to the fine-scale variations in $Q_\mu(i,u)$. As a result, the dependence of Q_μ on u may be lost when using instead the approximation $\tilde{Q}_\mu$.

point equation in the vector J:

$$(WH_\mu)(J) = J, \tag{54}$$

where $W : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a mapping (possibly nonlinear but independent of μ). Policy evaluation by solving the projected equation corresponds to $W = \Pi$. Rather than specify properties of H_μ under which H has a unique fixed point (as in [94, 95] and [96]), it is simpler for our purposes to introduce corresponding assumptions on the mappings W and WH_μ. In particular, we assume the following:

a) For each J, the minimum in (53) is attained, in the sense that there exists $\bar{\mu} \in \mathcal{M}$ such that $HJ = H_{\bar{\mu}}J$.

b) For each $\mu \in \mathcal{M}$, the mappings W and WH_μ are monotone in the sense that

$$WJ \leqslant W\bar{J},\ (WH_\mu)(J) \leqslant (WH_\mu)(\bar{J}),\ \forall J, \bar{J} \in \mathbb{R}^n \text{ with } J \leqslant \bar{J}. \tag{55}$$

c) For each μ, the solution $\tilde{J}_\mu$ of (54) is unique, and for all J such that $(WH_\mu)(J) \leqslant J$, we have

$$\tilde{J}_\mu = \lim_{k\to\infty} (WH_\mu)^k(J).$$

Based on condition a), we introduce a policy improvement operation that is similar to the case where $H_\mu = T_\mu$: the 'improved' policy $\bar{\mu}$ satisfies $H_{\bar{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$. Note that condition c) is satisfied if WH_μ is a contraction on S, while condition b) is satisfied if W is a matrix with nonnegative components and H_μ is monotone for all μ. These conditions also hold for the constrained policy iteration method (46), (47), where H_μ is the nonlinear mapping $\bar{T}_\mu$ of (49).

Proposition 6 Let conditions a)–c) hold. Consider the approximate policy iteration method that uses the fixed point $\tilde{J}_\mu$ of the mapping WH_μ for evaluation of the policy μ (54) and the equation $H_{\bar{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$ for policy improvement. Assume that the method is initiated with some policy in $\mathcal{M}$, and it is operated so that it terminates when a policy $\bar{\mu}$ is obtained such that $H_{\bar{\mu}}\tilde{J}_{\bar{\mu}} = H\tilde{J}_{\bar{\mu}}$. Then, the method terminates in a finite number of iterations, and the vector $\tilde{J}_{\bar{\mu}}$ obtained upon termination is a fixed point of WH.

Proof Similar to the standard proof of convergence of (exact) policy iteration, we use the policy improvement equation $H_{\bar{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$, the monotonicity of W, and the policy evaluation (54) to write

$$(WH_{\bar{\mu}})(\tilde{J}_\mu) = (WH)(\tilde{J}_\mu) \leqslant (WH_\mu)(\tilde{J}_\mu) = \tilde{J}_\mu.$$

By iterating with the monotone mapping $WH_{\bar{\mu}}$ and by using condition (c), we obtain

$$\tilde{J}_{\bar{\mu}} = \lim_{k\to\infty} (WH_{\bar{\mu}})^k(\tilde{J}_\mu) \leqslant \tilde{J}_\mu.$$

There are finitely many policies, so we must have $\tilde{J}_{\bar{\mu}} = \tilde{J}_\mu$ after a finite number of iterations, which, using the policy improvement equation $H_{\bar{\mu}}\tilde{J}_\mu = H\tilde{J}_\mu$, implies that $H_{\bar{\mu}}\tilde{J}_{\bar{\mu}} = H\tilde{J}_{\bar{\mu}}$. Thus, the algorithm terminates with $\bar{\mu}$, and since $\tilde{J}_{\bar{\mu}} = (WH_{\bar{\mu}})(\tilde{J}_{\bar{\mu}})$, it follows that $\tilde{J}_{\bar{\mu}}$ is a fixed point of WH.

Note the mechanism by which the preceding policy iteration method avoids the oscillations described in Section 3.5. Again, the policy improvement step yields $\bar{\mu}$ from μ if and only if $\tilde{J}_\mu \in R_{\bar{\mu}}$, where $R_{\bar{\mu}} = \{J | H_{\bar{\mu}}J = HJ\}$ is the corresponding set of the greedy partition (which is independent of the method used for policy evaluation). However, the monotonicity inherent in the policy evaluation step guarantees that

$$\tilde{J}_{\bar{\mu}} \leqslant \tilde{J}_\mu,\ \ \forall \mu, \bar{\mu} \in \mathcal{M} \text{ such that } \tilde{J}_\mu \in R_{\bar{\mu}},$$

so a policy oscillation is impossible. Note also that the preceding proof applies to the more general case where for policy improvement we use any policy $\bar{\mu}$ such that $H_{\bar{\mu}}\tilde{J}_\mu \leqslant \tilde{J}_\mu$, with strict inequality $H_{\bar{\mu}}\tilde{J}_\mu \neq \tilde{J}_\mu$ when $H_{\bar{\mu}}\tilde{J}_\mu \neq \tilde{J}_\mu$. This allows for approximate minimization over candidate policies in the policy improvement step.

An important special case where Proposition 6 applies and policies converge is when $H_\mu = T_\mu$, $H = T$, W is linear of the form $W = \Phi D$, where Φ and D are $n \times s$ and $s \times n$ matrices, respectively, whose rows are probability distributions, and the policy evaluation uses the linear feature-based approximation $\tilde{J}_\mu = \Phi r_\mu$. This is the case of policy evaluation by aggregation, which will be discussed in the next section. Then, it can be seen that W and WT_μ are monotone and that WT_μ is a sup-norm contraction (since W is nonexpansive with respect to the sup norm), so that conditions a)–c) are satisfied. The same is true in the more general case $W = \Phi D$, where the matrix W has nonnegative components, and its row sums are less than or equal to 1, i.e.,

$$\sum_{m=1}^{s} \Phi_{im} D_{mj} \geqslant 0,\ \ \forall i, j = 1, \ldots, n,$$
$$\sum_{m=1}^{s} \Phi_{im} \sum_{j=1}^{n} D_{mj} \leqslant 1,\ \ \forall i = 1, \ldots, n.$$

If Φ and D have nonnegative components, the first of these relations is automatically satisfied, while the second is equivalent to the set of n linear inequalities

$$\phi(i)'\zeta \leqslant 1,\ \ \forall i = 1, \ldots, n, \tag{56}$$

where $\phi(i)'$ is the ith row of Φ, and $\zeta \in \mathbb{R}^s$ is the column vector of row sums of D, i.e., the one that has components [16]

$$\zeta(m) = \sum_{j=1}^{n} D_{mj},\ \ \forall m = 1, \ldots, s.$$

Even in this more general case, the policy evaluation (54) can be solved by using simulation and low-order calculations (see Section 6.8 in [9]).

Given a 'good' choice of $\Phi \geqslant 0$, an interesting question is how to construct effective algorithms for parametric optimization of a nonnegative matrix D, subject to the constraints (56). This is similar to the basis function optimization issue mentioned in the introduction.

Finally, let us note that a special case when (56) may be easily satisfied arises when through a reordering of indexes, the matrix D can be partitioned in the form $D = [\Delta\ \ 0]$, where Δ is a positive definite diagonal matrix with diago-

[16] A column of Φ that has both positive and negative components may be replaced with the two columns that contain its positive and the opposite of its negative components. This will create a new nonnegative matrix Φ with as many as twice the number of columns, and will also enlarge the approximation subspace S (leading to no worse approximation). Then, the matrix D may be optimized subject to $D \geqslant 0$ and the constraints (56), with respect to some performance criterion.

nal elements d_m, $m = 1, \ldots, s$, satisfying

$$\sum_{m=1}^{s} \Phi_{im} d_m \leqslant 1, \ \ \forall i = 1, \ldots, n.$$

An example of such a structure arises in coarse grid discretization/aggregation schemes (see Section 4).

3.7 The case of hard aggregation

If the mapping H_μ is monotone in the sense that $H_\mu J \leqslant H_\mu \bar{J}$ for all $J, \bar{J} \in \mathbb{R}^n$ with $J \leqslant \bar{J}$ (as it typically is in DP models), then the monotonicity assumption (55) is satisfied if W is a projection matrix Π (the same for all μ), which satisfies $\Pi J \geqslant 0$ for all J with $J \geqslant 0$ (or equivalently $\Pi \mathbb{R}^n_+ = S \cap \mathbb{R}^n_+$, where $\mathbb{R}^n_+$ is the nonnegative orthant). This is so if and only if all the components of Π are nonnegative. A special case where this is true is when Φ has nonnegative components, and has linearly independent columns that are orthogonal with respect to the inner product $\langle x_1, x_2 \rangle = x_1' \Xi x_2$. This follows from the projection formula $\Pi = \Phi(\Phi' \Xi \Phi)^{-1} \Phi' \Xi$ and the fact that $\Phi' \Xi \Phi$ is positive definite and diagonal. An interesting special case is when policy evaluation is done by hard aggregation, one of the methods from the class to be discussed in the next section. Here, the state space $\{1, \ldots, n\}$ is partitioned in s nonempty subsets $I_1, \ldots, I_s$ and:

1) The ℓth column of Φ has components that are 1 or 0 depending on whether they correspond to an index in I_ℓ or not.

2) The ℓth row of D is a probability distribution $(d_{\ell 1}, \ldots, d_{\ell n})$ whose components are positive depending on whether they correspond to an index in I_ℓ or not, i.e., $\sum_{j=1}^{n} d_{\ell j} = 1$, $d_{\ell j} > 0$ if $j \in I_\ell$, and $d_{\ell j} = 0$ if $j \notin I_\ell$.

With these definitions of Φ and D, it can be verified that ΦD is given by the projection formula

$$\Phi D = \Phi(\Phi' \Xi \Phi)^{-1} \Phi' \Xi,$$

where Ξ is the diagonal matrix with the nonzero components of D along the diagonal. In fact, Π, the projection on S with respect to $\| \cdot \|_\xi$, can be written in the explicit form

$$(\Pi J)(i) = \sum_{j \in I_\ell} d_{\ell j} J(j), \ \ \forall i \in I_\ell, \ \ \ell = 1, \ldots, s.$$

Thus, assuming that D (and hence Π) is held constant, policies converge and the more favorable error bound (43) holds. [17]

3.8 Q-factor approximations, exploration, and optimistic policy iteration

The policy iteration methods with cost function approximation discussed so far rely on the calculation of the approximation Φr to the cost function of the current policy μ, which is then used for policy improvement using the minimization (41). Carrying out this minimization requires knowledge of the transition probabilities $p_{ij}(u)$ and calculation of the associated expected values for all controls $u \in U(i)$ (otherwise a time-consuming simulation of these expected values is needed). An interesting 'model-free' alternative is to perform approximate policy evaluation by computing approximate Q-factors

$$\tilde{Q}(i, u, r) \approx (F_\mu Q)(i, u), \ \ \forall (i, u), \tag{57}$$

where $F_\mu Q$ is the Q-learning mapping given by

$$(F_\mu Q)(i, u) = \sum_{j=1}^{n} p_{ij}(u) \big(g(i, u, j) + \alpha Q(j, \mu(j)) \big), \ \ \forall (i, u), \tag{58}$$

and to use the minimization

$$\bar{\mu}(i) \in \arg \min_{u \in U(i)} \tilde{Q}(i, u, r), \ \ \forall i, \tag{59}$$

for policy improvement. Here, r is an adjustable parameter vector and $\tilde{Q}(i, u, r)$ is a parametric architecture, possibly of the linear form

$$\tilde{Q}(i, u, r) = \sum_{k=1}^{s} r_k \phi_k(i, u),$$

where $\phi_k(i, u)$ are basis functions that depend on both state and control.

An important point is that given the current policy μ, we can construct Q-factor approximations for that policy using any method for constructing cost vector approximations. The way to do this is to apply the latter method to the Markov chain whose states are the pairs (i, u), and the probability of transition from (i, u) to (j, v) is

$$p_{(i,u)(j,v)} = p_{ij}(u) \delta\big(v = \mu(j)\big), \tag{60}$$

where as earlier $\delta(\cdot)$ denotes the indicator function: $\delta\big(v = \mu(j)\big) = 1$ if $v = \mu(j)$ and is equal to 0 otherwise. This is the probabilistic mechanism by which state-control pairs evolve under the stationary policy μ. However, a major concern here is that the state-control pairs (i, u) with $u \neq \mu(i)$ are never generated in this Markov chain, so they are not represented in the cost samples used to construct the approximation $\tilde{Q}(i, u, r)$. This creates an acute difficulty due to diminished exploration, which must be addressed in any simulation-based implementation.

To address the issue of exploration in evaluating (exactly or approximately) the Q-factors of a given policy, we may use the LSTD(λ) approach in conjunction with the algorithm (31)–(33) of Section 2.3. We recall that, in this algorithm, row and column sampling is done by generating a sequence of state-control pairs $\big\{(i_0, u_0), (i_1, u_1), \ldots\big\}$. The Q-factor approximation is $\Phi \hat{r}$, where $\hat{r}$ is the solution of the system $C_k^{(\lambda)} r = d_k^{(\lambda)}$, with $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$ generated recursively by

$$C_k^{(\lambda)} = \Big(1 - \frac{1}{k+1}\Big) C_{k-1}^{(\lambda)} + \frac{1}{k+1} z_k \Big(\phi(i_k, u_k) - \alpha \frac{p_{(i_k,u_k)(i_{k+1},u_{k+1})}}{\ell_{(i_k,u_k)(i_{k+1},u_{k+1})}} \phi(i_{k+1}, u_{k+1}) \Big)', \tag{61}$$

$$d_k^{(\lambda)} = \Big(1 - \frac{1}{k+1}\Big) d_{k-1}^{(\lambda)} + \frac{1}{k+1} z_k \alpha \frac{p_{(i_k,u_k)(i_{k+1},u_{k+1})}}{\ell_{(i_k,u_k)(i_{k+1},u_{k+1})}} g(i_k, u_k, i_{k+1}), \tag{62}$$

[17] Van Roy [47] has established especially favorable error bounds for a hard aggregation-based approximate DP method. However, the policy iteration method considered in [47] is not the same as the one considered here, because the corresponding matrix Π is not constant and policy convergence is not guaranteed. It is not clear whether it is practically advantageous to select Π in the manner suggested by Van Roy [47], compared to keeping it constant as in the standard hard aggregation scheme of Section 4.

where z_k is generated by

$$z_k = \lambda\alpha \frac{p_{(i_{k-1},u_{k-1})(i_k,u_k)}}{\ell_{(i_{k-1},u_{k-1})(i_k,u_k)}} z_{k-1} + \phi(i_k,u_k). \quad (63)$$

Here, $\phi(i,u)'$ is the row of Φ corresponding to (i,u), $p_{(i,u)(j,v)}$ are the transition probabilities between state-control pairs given by (60), and $\ell_{(i,u)(j,v)}$ are the corresponding transition probabilities governing row and column sampling. A particularly interesting choice is

$$\ell_{(i,u)(j,v)} = p_{ij}(u)\nu(v|j), \quad (64)$$

where ν is a randomized policy that affects exploration by mapping a state j to control v with probability $\nu(v|j)$. Then, from (60) and (64), we see that the ratios in algorithm (61)–(63) simplify

$$\frac{p_{(i,u)(j,v)}}{\ell_{(i,u)(j,v)}} = \frac{\delta\big(v=\mu(j)\big)}{\nu(v|j)},$$

(again, we adopt the convention $0/0=0$), so knowledge of $p_{ij}(u)$ is not needed, and the algorithm maintains its model-free character.

Recently, an alternative approach to Q-learning with exploration enhancement has been proposed, which is new even in the context of exact DP (Bertsekas and Yu [75]). It is based on replacing the Q-learning mapping F_μ of (58) with the mapping

$$\begin{aligned}&(F_{J,\nu}Q)(i,u)\\ &= \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)\\ &\quad +\alpha \sum_{v\in U(j)} \nu(v|j)\min\{J(j),Q(j,v)\}\big),\ \ \forall (i,u), \quad (65)\end{aligned}$$

which depends on a vector $J\in\mathbb{R}^n$, with components denoted $J(i)$, and on a randomized policy ν, which for each state j defines a probability distribution

$$\{\nu(v|j)|v\in U(j)\}$$

over the feasible controls at j, and may depend on the 'current policy' μ. Thus, solving a linear equation (the traditional policy evaluation method) is replaced by solving an optimal stopping problem: finding a fixed point of the mapping (65) (a similarity with the constrained policy iteration (46), (47) described earlier).

The exploration-enhanced policy iteration algorithm is similar to the ordinary version (57)–(59), except that it is based on the mapping $F_{J,\nu}$ in place of F_μ. Its policy evaluation phase is

$$\tilde{Q}(i,u,r) \approx (F_{J,\nu}Q)(i,u), \quad (66)$$

and its policy improvement phase is (59). At the end of an iteration, the vector J is set to

$$J(i) = \min_{u\in U(i)} (F_{J,\nu}\tilde{Q})(i,u),\ \ \forall i. \quad (67)$$

The policy ν may be chosen arbitrarily at each iteration. It may encode aspects of the 'current policy' μ but may also allow for arbitrary and easily controllable amount of exploration. For extreme choices of ν and a lookup table representation, the algorithm yields as special cases the classic Q-learning/value iteration and policy iteration methods. Together with linear Q-factor approximation, the algorithm may be combined with the TD(0)-like method of Tsitsiklis and Van Roy [76], which can be used to solve the associated stopping problems with low overhead per iteration, thereby resolving the issue of exploration.

The enhanced policy iteration algorithm also has some interesting properties in the context of exact/lookup table policy iteration: it admits asynchronous and stochastic iterative implementations, which can be attractive alternatives to standard methods of asynchronous/optimistic policy iteration and Q-learning. Its advantage is more robust convergence properties (it is not susceptible to the fragility of asynchronous policy iteration demonstrated by Williams and Baird [19] and of optimistic policy iteration with lookup table representation discussed by Section 5.4 in [1]); see Bertsekas and Yu [75,98] for further elaboration. An important fact regarding these properties is that, for every randomized policy ν, the mapping underlying the algorithm (66), (67),

$$(Q,J) \mapsto \left(F_{J,\nu}Q, MF_{J,\nu}Q\right), \quad (68)$$

where

$$(MF_{J,\nu}Q)(i) = \min_{u\in U(i)} (F_{J,\nu}Q)(i,u),\ \ \forall i,$$

is a sup-norm contraction of modulus α, and its unique fixed point is the optimal Q-factor and cost vector pair (Q^*,J^*).

The contraction property just described, and other similar properties, forms the basis for a variety of asynchronous stochastic Q-learning algorithms, which update Q-factors and relate to policy iteration with an optimistic character (a limited number of samples used to update Q for fixed J and ν, between updates of J). These algorithms may naturally involve enhanced exploration through the use of policies ν, which may relate but need not be identical to the policies under evaluation. We refer to Bertsekas and Yu [75] for a detailed discussion, convergence analysis, and also error bounds (in the case, where Q-factors are compactly represented through a set of basis functions).

The contraction property of the mapping (68) may also be used to construct deterministic asynchronous policy iteration algorithms with an optimistic character, which update costs (rather than Q-factors); see [75,98]. In these algorithms, we use deterministic policies μ^k in place of ν, and for each state i, there is a set of times $\mathcal{T}(i)$ at which we do a 'policy evaluation' iteration at i, given by

$$\begin{aligned}V^{k+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^k(i)\big)\big(g\big(i,\mu^k(i),j\big)\\ +\alpha\min\{J^k(j),V^k(j)\}\big),\end{aligned}$$

and we leave $J^k(i)$, $\mu^k(i)$ unchanged. There is also an infinite set of times $\mathcal{T}(i)$ [disjoint from $\mathcal{T}(i)$] at which we do a 'policy improvement' at i, given by

$$\begin{aligned}J^{k+1}(i) &= V^{k+1}(i)\\ &= \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)\\ &\quad +\alpha\min\{J^k(j),V^k(j)\}\big),\end{aligned}$$

while setting $\mu^{k+1}(i)$ to a u that attains the minimum above. Based on the sup-norm contraction property of mapping (68), it can be shown that this algorithm satisfies $\lim_{k\to\infty} J^k(i) = \lim_{k\to\infty} V^k(i) = J^*(i)$ for all i, with arbitrary initial conditions (J^0,V^0,μ^0), thus overcoming the convergence difficulties demonstrated by Williams and Baird [19] for the natural version of asynchronous policy iteration.

4 Policy evaluation by aggregation

In this section, we consider policy iteration with aggregation-based approximate policy evaluation. The aggregation approach resembles the projected equation approach because it also constructs cost approximations of the form Φr. However, there are some differences, the most important of which is that in aggregation methods there are restrictions in the form of Φ, so there is less freedom in choosing the approximation subspace. Moreover, from a mathematical point of view, in the aggregation approach, there are no projections with respect to Euclidean norms, the simulations can be done more flexibly, and the underlying contractions are with respect to the sup-norm rather than a Euclidean norm.

From a conceptual point of view, the most important difference from the projected equation approach is that aggregation is based on problem approximation (rather than Bellman equation approximation): the original problem is approximated with a related 'aggregate' problem, which is then solved exactly to yield a cost-to-go approximation for the original problem. As a result, policy iteration with the aggregation approach is not afflicted by some of the pathologies that emanate from compact feature-based representations of cost functions, such as policy oscillations and chattering. The sequence of generated policies converges (to an optimal policy of the aggregate problem) with monotonic cost improvement (for the aggregate problem), and a more favorable error bound can be derived than for the projected equation case.

To construct an aggregation framework for the discounted MDP of the preceding section, we introduce a finite set $\mathcal{A}$ of aggregate states, and we introduce two sets of probabilities, which relate the original system states with the aggregate states:

1) For each aggregate state x and original system state i, we specify the disaggregation probability d_{xi} (we have $\sum_{i=1}^{n} d_{xi} = 1$ for each $x \in \mathcal{A}$). Roughly, d_{xi} may be interpreted as the 'degree to which x is represented by i'.

2) For each aggregate state y and original system state j, we specify the aggregation probability ϕ_{jy} (we have $\sum_{y\in\mathcal{A}} \phi_{jy} = 1$ for each $j = 1, \ldots, n$). Roughly, ϕ_{jy} may be interpreted as the 'degree of membership of j in the aggregate state y'. The vectors $\{\phi_{jy} | j = 1, \ldots, n\}$ may also be viewed as basis functions that will be used to provide a cost approximation for the original problem.

We mention a few examples, and we refer to [18] Vol. I, Section 6.3, and to [9] for further discussion and examples.

a) In hard and soft aggregation, we group the original system states into subsets, and we view each subset as an aggregate state. In hard aggregation, each state belongs to one and only one subset, and the aggregation probabilities are

$$\phi_{jy} = 1 \quad \text{if system state } j \text{ belongs to aggregate state/subset } y.$$

The disaggregation probabilities could be

$$d_{xi} = 1/n_x \quad \text{if system state } i \text{ belongs to aggregate state/subset } x,$$

where n_x is the number of states of x (this implicitly assumes that all states that belong to aggregate state/subset y are 'equally representative'). In soft aggregation, we allow the aggregate states/subsets to overlap, with the aggregation probabilities ϕ_{jy} quantifying the 'degree of membership' of j in the aggregate state/subset y.

b) In various discretization schemes, each original system state j is associated with a convex combination of aggregate states:

$$j \sim \sum_{y\in\mathcal{A}} \phi_{jy} y,$$

for some nonnegative weights ϕ_{jx}, whose sum is 1, and which are viewed as aggregation probabilities (this makes geometrical sense if both the original and the aggregate states are associated with points in a Euclidean space).

c) In coarse grid schemes, a subset of representative states is chosen, each being an aggregate state. Thus, each aggregate state x is associated with a unique original state i_x, and we may use the disaggregation probabilities $d_{xi} = 1$ for $i = i_x$ and $d_{xi} = 0$ for $i \neq i_x$. The aggregation probabilities are chosen as in the preceding case b).

The aggregation approach approximates the cost vector of a policy with Φr, where $r \in \mathbb{R}^s$ is a weight vector to be determined, and Φ is the matrix whose jth row consists of the aggregation probabilities $\phi_{j1}, \ldots, \phi_{js}$. Thus, aggregation involves an approximation architecture similar to the one of projected equation methods: it uses as features the aggregation probabilities. There is a reciprocal possibility: starting from a set of s features for each state, we may construct a feature-based hard aggregation scheme by grouping together states with 'similar features'. In particular, we may use a more or less regular partition of the feature space, which induces a possibly irregular partition of the original state space into aggregate states (all states whose features fall in the same set of the feature partition form an aggregate state). This is a general approach for passing from a feature-based approximation of the cost vector to an aggregation-based approximation (an interesting special case is when one of the features is a vector that approximates the cost vector of some known suboptimal policy). Unfortunately, in the resulting aggregation scheme, the number of aggregate states may become very large.

The aggregation and disaggregation probabilities specify a dynamical system involving both aggregate and original system states. In this system:

i) From aggregate state x, we generate original system state i according to d_{xi}.

ii) We generate transitions from original system state i to original system state j according to $p_{ij}(u)$, with cost $g(i, u, j)$.

iii) From original system state j, we generate aggregate state y according to ϕ_{jy}.

Let us now introduce the vectors $\tilde{J}_0$, $\tilde{J}_1$, and R^* where $R^*(x)$ is the optimal cost-to-go from aggregate state x; $\tilde{J}_0(i)$ is the optimal cost-to-go from original system state i that has just been generated from an aggregate state (left side of Fig. 4); and $\tilde{J}_1(j)$ is the optimal cost-to-go from original system state j that has just been generated from an original system state (right side of Fig. 4).

Note that because of the intermediate transitions to aggregate states, $\tilde{J}_0$ and $\tilde{J}_1$ are different.

These three vectors satisfy the following three Bellman's equations:

$$R^*(x) = \sum_{i=1}^{n} d_{xi}\tilde{J}_0(i), \quad x \in \mathcal{A},$$

$$\tilde{J}_0(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha\tilde{J}_1(j)\big), \quad i = 1,\dots,n,$$

$$\tilde{J}_1(j) = \sum_{y\in\mathcal{A}} \phi_{jy}R^*(y), \quad j = 1,\dots,n.$$

By combining these equations, we obtain an equation for R^*:

$$R^*(x) = (FR^*)(x), \quad x \in \mathcal{A},$$

where F is the mapping defined by

$$(FR)(x) = \sum_{i=1}^{n} d_{xi} \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \sum_{y\in\mathcal{A}} \phi_{jy}R(y)\big), \quad x \in \mathcal{A}. \tag{69}$$

It can be seen that F is a sup-norm contraction mapping and has R^* as its unique fixed point. This follows from standard contraction arguments and the fact that d_{xi}, $p_{ij}(u)$, and ϕ_{jy} are all transition probabilities. [18]

Once R^* is found, the optimal cost-to-go of the original problem may be approximated by $\tilde{J}_1 = \Phi R^*$, and a suboptimal policy may be found through the minimization defining $\tilde{J}_0$. Again, the optimal cost function approximation $\tilde{J}_1$ is a linear combination of the columns of Φ, which may be viewed as basis functions.

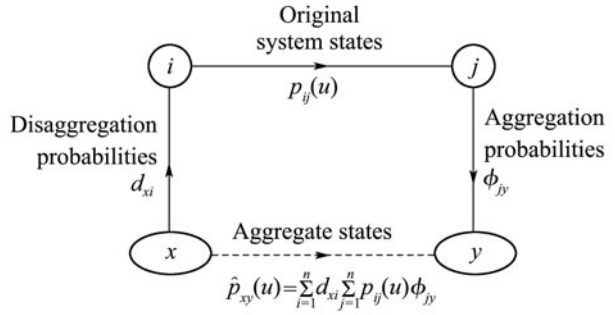

Fig. 4 Illustration of the transition mechanism of a dynamical system involving both aggregate and original system states.

4.1 Aggregation-based policy iteration

A policy iteration algorithm to find R^* may be constructed using aggregation for policy evaluation. It starts with a stationary policy μ^0 for the original problem, and given μ^k, it finds R_{μ^k} satisfying $R_{\mu^k} = F_{\mu^k}R_{\mu^k}$, where F_μ is the mapping defined by

$$(F_\mu R)(x) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(\mu(i))\Big(g(i,\mu(i),j) + \alpha \sum_{y\in\mathcal{A}} \phi_{jy}R_\mu(y)\Big), \quad x \in \mathcal{A} \tag{70}$$

(this is the policy evaluation step). It then generates μ^{k+1} by

$$\mu^{k+1}(i) \in \arg\min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \sum_{y\in\mathcal{A}} \phi_{jy}R_{\mu^k}(y)\Big), \quad \forall i \tag{71}$$

(this is the policy improvement step). Using the analysis of Section 3.6 and Proposition 6 (with W equal to the monotone mapping ΦD), it can be shown that this policy iteration algorithm terminates finitely to some policy corresponding to the unique fixed point of F. The key fact here is that F and F_μ are not only sup-norm contractions, but also have the monotonicity property of DP mappings, which is used in an essential way in the convergence proof of ordinary policy iteration.

We may implement the preceding policy iteration method by simulation using either a matrix inversion or an iterative approach. For a given policy μ, the aggregate version of Bellman's equation, $R = F_\mu R$, is linear of the form (70)

$$R = DT_\mu(\Phi R),$$

where T_μ is the DP mapping associated with μ. We can thus write this equation as

$$ER = f,$$

where

$$E = I - \alpha DP\Phi, \quad f = Dg, \tag{72}$$

in analogy with the corresponding matrix and vector for the projected equation.

We may use low-dimensional simulation to approximate E and f based on a given number of samples, similar to (20) in Section 2. In particular, a sample sequence $\big\{(i_0,j_0),(i_1,j_1),\dots\big\}$ is obtained by first generating a sequence of states $\{i_0,i_1,\dots\}$ by sampling according to a distribution $\{\xi_i|i=1,\dots,n\}$ (with $\xi_i > 0$ for all i), and then by generating for each t the column index j_t using sampling according to the distribution $\{p_{i_t j}|j=1,\dots,n\}$. Given the first $k+1$ samples, we form the matrix $\hat{E}_k$ and vector $\hat{f}_k$ given by

$$\begin{cases} \hat{E}_k = I - \dfrac{\alpha}{k+1}\displaystyle\sum_{t=0}^{k} \frac{1}{\xi_{i_t}} d(i_t)\phi(j_t)', \\ \hat{f}_k = \dfrac{1}{k+1}\displaystyle\sum_{t=0}^{k} \frac{1}{\xi_{i_t}} d(i_t)g\big(i_t,\mu(i_t),j_t\big), \end{cases} \tag{73}$$

where $d(i)$ is the ith column of D and $\phi(j)'$ is the jth row of Φ. The convergence $\hat{E}_k \to E$ and $\hat{f}_k \to f$ follows from the expressions

$$E = I - \alpha \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)d(i)\phi(j)',$$

$$f = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)d(i)g(i,\mu(i),j),$$

the relation

$$\lim_{k\to\infty} \sum_{t=0}^{k} \frac{\delta(i_t=i,j_t=j)}{k+1} = \xi_i p_{ij},$$

and law of large numbers arguments (cf. Section 2).

[18] A quick proof is to observe that F is the composition $F = DT\Phi$, where T is the usual DP mapping, and D and Φ are the matrices with rows the disaggregation and aggregation distributions, respectively. Since T is a contraction with respect to the sup-norm $\|\cdot\|_\infty$, and D and Φ are sup-norm nonexpansive in the sense

$$\|Dx\|_\infty \leqslant \|x\|_\infty, \ \forall x \in \mathbb{R}^n, \quad \|\Phi y\|_\infty \leqslant \|y\|_\infty, \ \forall y \in \mathbb{R}^s,$$

it follows that F is a sup-norm contraction.

It is important to note that the sampling probabilities ξ_i are restricted to be positive, but are otherwise arbitrary and need not depend on the current policy. Moreover, their choice does not affect the obtained approximate solution of the equation $ER = f$. Because of this possibility, the problem of exploration is less acute in the context of policy iteration when aggregation is used for policy evaluation. This is in contrast with the projected equation approach, where the choice of ξ_i affects the projection norm and the solution of the projected equation as well as the contraction properties of the mapping ΠT.

Note also that, instead of using the probabilities ξ_i to sample original system states, we may alternatively sample the aggregate states x according to a distribution $\{\zeta_x | x \in \mathcal{A}\}$, generate a sequence of aggregate states $\{x_0, x_1, \ldots\}$, and then generate a state sequence $\{i_0, i_1, \ldots\}$ using the disaggregation probabilities. In this case, $\xi_i = \sum_{x \in \mathcal{A}} \zeta_x d_{xi}$ and (73) should be modified as follows:

$$\hat{E}_k = I - \frac{\alpha}{k+1} \sum_{t=0}^{k} \frac{m}{d_{x_t i_t}} d(i_t)\phi(j_t)',$$

$$\hat{f}_k = \frac{1}{k+1} \sum_{t=0}^{k} \frac{m}{d_{x_t i_t}} d(i_t) g\big(i_t, \mu(i_t), j_t\big),$$

where m is the number of aggregate states.

The corresponding LSTD-type method generates $\hat{R}_k = \hat{E}_k^{-1} \hat{f}_k$ and approximates the cost vector of μ with the vector $\Phi \hat{R}_k$:

$$\tilde{J}_\mu = \Phi \hat{R}_k.$$

There is also a regression-based version that is suitable for the case where $\hat{E}_k$ is nearly singular (cf. Section 2), as well as an iterative regression-based version of LSTD, which may be viewed as a special case of the (scaled) LSPE-type method (39), (40). The latter method takes the form

$$\hat{R}_{k+1} = (\hat{E}_k' \Sigma_k^{-1} \hat{E}_k + \beta I)^{-1} (\hat{E}_k' \Sigma_k^{-1} \hat{f}_k + \beta \hat{R}_k), \quad (74)$$

where $\beta > 0$ and Σ_k is a positive definite symmetric matrix. Note that, contrary to the projected equation case, for a discount factor $\alpha \approx 1$, $\hat{E}_k$ will always be nearly singular [since $DP\Phi$ is a transition probability matrix, (72). However, the iteration (74) is valid even if $\hat{E}_k$ is singular.

As noted earlier, the nonoptimistic version of the aggregation-based policy iteration method does not exhibit the oscillatory behavior of the one based on the projected equation approach (cf. Section 3.6). The generated policies converge since by aggregation, we are essentially solving exactly the DP problem associated with the aggregate system of Fig. 4. The convergence of policies in turn implies that the error bound (42) holds. This suggests a more regular behavior and an advantage in terms of approximation quality of aggregation-based approximate policy iteration relative to its projected equation-based counterpart.

A related property holds for optimistic policy iteration methods. They behave similar to optimistic policy iteration for the DP problem associated with the aggregate system of Fig. 4, and they do not exhibit the chattering phenomenon described in Section 3.5.

In conclusion the aggregation approach holds an advantage over the projected equation approach in terms of regularity of behavior, error guarantees, and exploration-related difficulties. Note, however, that the basis functions in the aggregation approach are restricted by the requirement that the rows of Φ must be probability distributions. For example, in the case of a single basis function ($s = 1$), there is only one possible choice for Φ in the aggregation context, namely the matrix whose single column is the unit vector. In practice this means that the number of columns of Φ may be much larger than in the projected equation approach.

4.2 Multistep aggregation

The aggregation methodology of this section can be generalized by considering a multistep aggregation-based dynamical system. This system, illustrated in Fig. 5, is specified by disaggregation and aggregation probabilities as before but involves $k > 1$ transitions between original system states in between transitions from and to aggregate states.

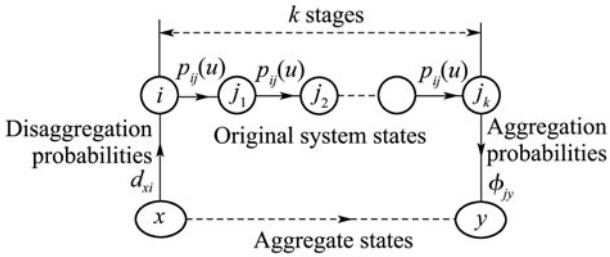

Fig. 5 The transition mechanism for multistep aggregation. It is based on a dynamical system involving aggregate states, and k transitions between original system states in between transitions from and to aggregate states.

We introduce vectors $\tilde{J}_0, \tilde{J}_1, \ldots, \tilde{J}_k$, and R^* where:

$R^*(x)$ is the optimal cost-to-go from aggregate state x.

$\tilde{J}_0(i)$ is the optimal cost-to-go from original system state i that has just been generated from an aggregate state (left side of Fig. 5).

$\tilde{J}_1(j_1)$ is the optimal cost-to-go from original system state j that has just been generated from an original system state i.

$\tilde{J}_m(j_m)$, $m = 2, \ldots, k$, is the optimal cost-to-go from original system state j_m that has just been generated from an original system state j_{m-1}. These vectors satisfy the following set of Bellman equations:

$$R^*(x) = \sum_{i=1}^{n} d_{xi} \tilde{J}_0(i), \quad x \in \mathcal{A},$$

$$\tilde{J}_0(i) = \min_{u \in U(i)} \sum_{j_1=1}^{n} p_{ij_1}(u)\big(g(i,u,j_1) + \alpha \tilde{J}_1(j_1)\big), \quad i = 1, \ldots, n, \quad (75)$$

$$\tilde{J}_m(j_m) = \min_{u \in U(j_m)} \sum_{j_{m+1}=1}^{n} p_{j_m j_{m+1}}(u)\big(g(j_m, u, j_{m+1}) + \alpha \tilde{J}_{m+1}(j_{m+1})\big), \quad j_m = 1, \ldots, n, \quad m = 1, \ldots, k-1, \quad (76)$$

$$\tilde{J}_k(j_k) = \sum_{y \in \mathcal{A}} \phi_{j_k y} R^*(y), \quad j_k = 1, \ldots, n. \quad (77)$$

By combining these equations, we obtain an equation for R^*:

$$R^*(x) = (FR^*)(x), \quad x \in \mathcal{A},$$

where F is the mapping defined by

$$FR = DT^k(\Phi R),$$

where T is the usual DP mapping of the problem. As earlier, it can be seen that F is a sup-norm contraction, but its contraction modulus is α^k rather than α.

There is a similar mapping corresponding to a fixed policy and it can be used to implement a policy iteration algorithm, which evaluates a policy through calculation of a corresponding vector R and then improves it. However, there is a major difference from the single-step aggregation case: a policy involves a set of k control functions $\{\mu_0, \ldots, \mu_{k-1}\}$, and while a known policy can be easily simulated, its improvement involves multistep lookahead using the minimizations of (75)–(77), and may be costly. Thus, multistep aggregation is a useful idea only for problems where the cost of this multistep lookahead minimization (from a single given starting state) is not prohibitive. By contrast, policy improvement using projected equation-based multistep policy evaluation methods such as TD(λ) does not have this limitation. On the other hand, note that from the theoretical point of view, a multistep scheme provides a means of better approximation of the true optimal cost vector J^* independent of the use of a large number of aggregate states. This can be seen from (75)–(77), which based on classical value iteration convergence results, show that $\tilde{J}_0(i) \to J^*(i)$ as $k \to \infty$ regardless of the choice of aggregate states.

4.3 Distributed asynchronous aggregation

We now consider the distributed solution of discounted DP problems using aggregation. We envision a network of s processors/agents, with each processor updating asynchronously a local policy and a local cost function, defined on a subset of the state space. Aggregate estimates of the computed values are communicated asynchronously between processors and are used to perform local cost updates and possibly local policy updates. In extreme cases, the aggregate estimates available to the agents may be greatly outdated, or otherwise may carry marginally useful information, thereby approaching a multiagent system model where each agent makes decisions based solely on local information.

In a synchronous value iteration method, each processor $a = 1, \ldots, s$, maintains/updates a (local) cost $J_a(i)$ for every state i in a subset of states I_a and also an aggregate cost

$$R_a = \sum_{i \in I_a} d_{ai} J_a(i),$$

where $\{d_{ai} | i \in I_a\}$ is a probability distribution. We assume that the subsets I_a, $a = 1, \ldots, s$, form a partition of the state space (as in hard aggregation, cf. Section 3.7), and we generically denote by J and R the vectors with components $J_a(i)$, $i \in I_a$, and R_a, $a = 1, \ldots, s$, respectively. Note that J is an n-dimensional vector, the Cartesian product $J = \prod_{a=1,\ldots,s} J_a$. For state i, $J(i)$ is the ith component of the vector J_a corresponding to the processor a for which $i \in I_a$.

We first discuss a value iteration algorithm, which involves updating both R and J according to

$$R_a^t = \sum_{i \in I_a} d_{ai} J_a^t(i), \ \ \forall a = 1, \ldots, s, \tag{78}$$

$$J_a^{t+1}(i) = \min_{u \in U(i)} H_a(i, u, J^t, R^t), \ \ \forall i \in I_a, \ a = 1, \ldots, s, \tag{79}$$

where the mapping H_a is defined for all $a = 1, \ldots, s$, $i \in I_a$, $u \in U(i)$, and $J \in \mathbb{R}^n$, $R \in \mathbb{R}^s$, by

$$H_a(i, u, J, R) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j) + \alpha \sum_{j \in I_a} p_{ij}(u) J(j) + \alpha \sum_{j \notin I_a} p_{ij}(u) R_{a(j)}, \tag{80}$$

and where for each original system state j, we denote by $a(j)$ the subset to which j belongs (i.e., $j \in I_{a(j)}$). Thus, the iteration (79) is the same as ordinary value iteration, except that the aggregate costs $R_{a(j)}$ are used for the states $j \notin I_a$, whose costs are updated by other processors. It is possible to show that the iteration (78), (79) involves a sup-norm contraction mapping of modulus α, so it converges to the unique solution of the following system of equations in (J, R)

$$\begin{cases} R_a = \sum_{i \in I_a} d_{ai} J_a(i), \ \ \forall a = 1, \ldots, s, \\ J_a(i) = \min_{u \in U(i)} H_a(i, u, J, R), \ \ \forall i \in I_a \end{cases} \tag{81}$$

(The contraction property follows from the fact that $\{d_{ai} | i \in I_a\}$ is a probability distribution.)

In the algorithm (78), (79), all processors a must be updating their aggregate costs R_a and local costs $J_a(i)$ synchronously and communicate the aggregate costs to the other processors before a new iteration may begin. In a more practical asynchronous version of the method, the aggregate costs R_a may be outdated because of communication 'delays' between processors. In this case, the iteration (78), (79) is modified to take the form

$$J_a^{t+1}(i) = \min_{u \in U(i)} H_a\big(i, u, J^t, R_1^{\tau_{a1}(t)}, \ldots, R_s^{\tau_{as}(t)}\big), \quad \forall i \in I_a, \ a = 1, \ldots, s, \tag{82}$$

with $0 \leqslant \tau_{aa'}(t) \leqslant t$ for $a, a' = 1, \ldots, s$, and

$$R_a^\tau = \sum_{i \in I_a} d_{ai} J_a^\tau(i), \ \ \forall a = 1, \ldots, s.$$

The differences $t - \tau_{aa'}(t)$, $a, a' = 1, \ldots, s$, in (82) may be viewed as 'delays' between the current time k and the times $\tau_{aa'}(t)$ when the corresponding aggregate costs were computed by other processors. Moreover, the iteration (82) may be executed only at a subset $\mathcal{T}_a$ of times for each processor a, whereas for the other times $t \notin \mathcal{T}_a$ the vector J_a remains unchanged, i.e., $J_a^{t+1}(i) = J_a^t(i)$ for all $i \in I_a$. The convergence of this algorithm to the unique solution of (81) can be shown using the theory of asynchronous distributed DP developed in Bertsekas [99], and has been discussed recently in the paper by Bertsekas and Yu [98].

An asynchronous policy iteration method involving aggregation operates similarly, except that each processor a performs a policy evaluation iteration at a subset of times $t \in \mathcal{T}_a$, a policy improvement iteration at a subset of times $t \in \bar{\mathcal{T}}_a \subset \mathcal{T}_a$, and no update at the remaining times. The local policy improvement iteration is

$$\mu_a^{t+1}(i) = \begin{cases} \arg \min_{u \in U(i)} H_a\big(i, u, J^t, R_1^{\tau_{a1}(t)}, \ldots, R_m^{\tau_{am}(t)}\big) \\ \qquad \text{if } t \in \bar{\mathcal{T}}_a, \\ \mu_a^t(i) \text{ if } t \notin \bar{\mathcal{T}}_a, \end{cases} \quad \forall i \in I_a. \tag{83}$$

The local policy evaluation is iterative/optimistic according to

$$J_a^{t+1}(i) = \begin{cases} H_a\big(i, \mu_a^t(i), J^t, R_1^{\tau_{a1}(t)}, \ldots, R_s^{\tau_{as}(t)}\big) \\ \qquad \text{if } t \in \mathcal{T}_a, \\ J_a^t(i) \text{ if } t \notin \mathcal{T}_a, \end{cases} \quad \forall i \in I_a. \quad (84)$$

Unfortunately, this algorithm can fail even under favorable circumstances, as shown by Williams and Baird [19]: a lookup table representation (one processor for each state), only one processor updating at a time, and a deterministic system. An alternative and almost as simple algorithm was proposed recently in [98], which was shown to converge to the optimum under the most general conditions. In this algorithm, the iteration (84) is replaced by

$$J_a^{t+1}(i) = \begin{cases} \min\big\{V_a^t(i), H_a\big(i, \mu_a^t(i), J^t, R_1^{\tau_{a1}(t)}, \ldots, R_s^{\tau_{as}(t)}\big)\big\} \\ \qquad \text{if } t \in \mathcal{T}_a, \\ J_a^t(x) \text{ if } t \notin \mathcal{T}_a, \end{cases} \quad \forall i \in I_a. \quad (85)$$

where $V_a^t(i)$ is the value of the minimum computed during the latest policy improvement iteration (83):

$$V_a^{t+1}(i) = \begin{cases} \min\limits_{u \in U(i)} H_a\big(i, u, J^t, R_1^{\tau_{a1}(t)}, \ldots, R_s^{\tau_{as}(t)}\big) \\ \qquad \text{if } t \in \bar{\mathcal{T}}_a, \\ V_a^t(i) \text{ if } t \notin \bar{\mathcal{T}}_a, \end{cases} \quad \forall i \in I_a. \quad (86)$$

Other related algorithms, similarly convergent to the unique solution of (81), have also been discussed in [98].

5 Conclusions

We have surveyed some aspects of policy iteration methods with cost function approximation. From an analytical point of view, this is a subject with a rich theory and interesting algorithmic issues. From a practical point of view, this is a methodology that can address very large and difficult problems, and yet present major technical difficulties (such as exploration deficiencies and nearly singular equations) and exhibit unpredictable behaviors (such as policy oscillations and chattering), which are not fully understood at present and may seriously challenge practitioners.

We have contrasted two policy evaluation approaches: projected equation and aggregation. It appears that aggregation-based methods have more regular behavior, offer better error bound guarantees, and have less exploration-related difficulties than projected equation-based methods. On the other hand, aggregation methods are restricted in the choice of basis functions that they can use, and this can be a significant limitation for many problems.

Acknowledgements

Many thanks are due to Huizhen (Janey) Yu for extensive helpful discussions and suggestions.

References

[1] D. P. Bertsekas, H. Yu. *Solution of Large Systems of Equations Using Approximate Dynamic Programming Methods*. Report LIDS-P-2754. Cambridge: Laboratory for Information and Decision Systems, MIT, 2007.

[2] R. S. Sutton, A. G. Barto. *Reinforcement Learning*. Cambridge: MIT Press, 1998.

[3] A. Gosavi. *Simulation-Based Optimization Parametric Optimization Techniques and Reinforcement Learning*. New York: Springer-Verlag, 2003.

[4] X. R. Cao. *Stochastic Learning and Optimization: A Sensitivity-Based Approach*. New York: Springer-Verlag, 2007.

[5] H. Chang, M. Fu, J. Hu, et al. *Simulation-Based Algorithms for Markov Decision Processes*. New York: Springer-Verlag, 2007.

[6] S. Meyn. *Control Techniques for Complex Networks*. New York: Cambridge University Press, 2007.

[7] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: John Wiley & Sons, 2007.

[8] L. Busoniu, R. Babuska, B. De Schutter, et al. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. New York: CRC Press, 2010.

[9] D. P. Bertsekas. Approximate dynamic programming. Web-based chapter. *Dynamic Programming and Optimal Control*. 3rd ed. Belmont, MA: Athena Scientific, 2010: 321 – 540.

[10] D. White, D. Sofge. *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold, 1992.

[11] J. Si, A. Barto, W. Powell, et al, eds. *Learning and Approximate Dynamic Programming*. New York: IEEE, 2004.

[12] F. L. Lewis, G. G. Lendaris, D. Liu. Special issue on adaptive dynamic programming and reinforcement learning in feedback control. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 2008, 38(4): 896 – 897.

[13] A. G. Barto, S. J. Bradtke, S. P. Singh. Real-time learning and control using asynchronous dynamic programming. *Artificial Intelligence*, 1995, 72(1/2): 81 – 138.

[14] V. S. Borkar. Reinforcement learning: a bridge between numerical methods and Monte Carlo. *Perspectives in Mathematical Science – I: Probability and Statistics*, World Scientific Publishing Co., 2009: 71 – 91.

[15] F. L. Lewis, D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 2009, 9(3): 32 – 50.

[16] C. Szepesvari. *Reinforcement Learning Algorithms for MDPs*. Report TR09-13. Edmonton, CA: Department of Computing Science, University of Alberta, 2009.

[17] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons, 1994.

[18] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd ed. Belmont, MA: Athena Scientific, 2007.

[19] R. J. Williams, L. C. Baird. *Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-critic Learning Systems*. Report NU-CCS-93-11, Boston, MA: College of Computer Science, Northeastern University, 1993.

[20] I. Menache, S. Mannor, N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals Operation Research*, 2005, 134(1): 215 – 238.

[21] H. Yu, D. P. Bertsekas. Basis function adaptation methods for cost approximation in MDP. *Proceedings of 2009 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, New York: IEEE, 2009: 74 – 81.

[22] L. Busoniu, D. Ernst, B. De Schutter, et al. Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 2010, 41(1): 1 – 14.

[23] D. D. Castro, S. Mannor. *Adaptive Bases for Reinforcement Learning*. Berlin: Springer-Verlag, 2010.

[24] D. P. Bertsekas, H. Yu. Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 2009, 227(1): 27 – 50.

[25] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 1959, 3(3): 210 – 229.

[26] A. L. Samuel. Some studies in machine learning using the game of checkers – II: recent progress. *IBM Journal of Research and Development*, 1967, 11(6): 601 – 617.

[27] A. G. Barto, R. S. Sutton, C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1983, 13(5): 835 – 846.

[28] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988, 3(1): 9 – 44.

[29] G. J. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 1992, 8(3/4): 257 – 277.

[30] L. Gurvits, L. J. Lin, S. J. Hanson. *Incremental Learning of Evaluation Functions for Absorbing Markov Chains: New Methods and Theorems*. Princeton: Siemens Corporate Research, 1994.

[31] T. Jaakkola, M. I. Jordan, S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 1994, 6(6): 1185 – 1201.

[32] F. Pineda. Mean-field analysis for batched TD(λ). *Neural Computation*, 1997, 9: 1403 – 1419.

[33] J. N. Tsitsiklis, B. V. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 1997, 42(5): 674 – 690.

[34] J. N. Tsitsiklis, B. Van Roy. Average cost temporal-difference learning. *Automatica*, 1999, 35(11): 1799 – 1808.

[35] S. J. Bradtke, A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 1996, 22(1/3): 33 – 57.

[36] D. P. Bertsekas, S. Ioffe. *Temporal Differences-Based Policy Iteration and Applications in Neuro-dynamic Programming*. Report LIDS-P-2349. Cambridge: Laboratory for Information and Decision Systems, MIT, 1996.

[37] M. A. Krasnoselskii, J. B. Rutitcki, V. J. Stecenko, et al. *Approximate Solution of Operator Equations*. Translated by D. Louvish, Groningen: Wolters-Noordhoff Publisher, 1972.

[38] C. A. J. Fletcher. *Computational Galerkin Methods*. New York: Springer-Verlag, 1984.

[39] D. P. Bertsekas. *Projected Equations, Variational Inequalities, and Temporal Difference Methods*. Report LIDS-P-2808. Cambridge: Laboratory for Information and Decision Systems, MIT, 2009.

[40] H. Yu, D. P. Bertsekas. Error bounds for approximations from projected linear equations. *Mathematics of Operations Research*, 2010, 35(2): 306 – 329.

[41] H. Yu. *Least Squares Temporal Difference Methods: An Analysis Under General Conditions*. Technical report C-2010-39, Finland: Department Computer Science, University of Helsinki, 2010.

[42] H. Yu. Convergence of least squares temporal difference methods under general conditions. *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010: 1207 – 1214.

[43] S. P. Singh, T. Jaakkola, M. I. Jordan. Learning without state-estimation in partially observable Markovian decision processes. *Proceedings of the 11th Machine Learning Conference*, San Francisco, CA: Morgan Kaufmann Publishers Inc., 1994: 284 – 292.

[44] S. P. Singh, T. Jaakkola, M. I. Jordan. Reinforcement learning with soft state aggregation. *Advances in Neural Information Processing Systems 7*, Vancouver, BC, Canada, 1995: 361 – 368.

[45] G. J. Gordon. Stable function approximation in dynamic programming. *Machine Learning: Proceedings of the 12th International Conference*, Pittsburgh, PA: mCarnegie Mellon University, 1995.

[46] J. N. Tsitsiklis, B. V. Roy. Feature-based methods for large-scale dynamic programming. *Machine Learning*, 1996, 22(1/3): 59 – 94.

[47] B. Van Roy. Performance loss bounds for approximate value iteration with state aggregation. *Mathematics of Operations Research*, 2006, 31(2): 234 – 244.

[48] B. Scherrer. Should one compute the temporal difference fixed point or minimize the Bellman residual: the unified oblique projection view. *Proceedings of International Conference on Machine Learning*, Haifa, Israel, 2010: 959 – 966.

[49] C. Thiery, B. Scherrer. Least-squares policy iteration: bias-variance trade-off in control problems. *Proceedings of 27th International Conference on Machine Learning*, Haifa, Israel. 2010: 1071 – 1078.

[50] M. Wang, N. Polydorides, D. P. Bertsekas. *Approximate Simulation-Based Solution of Large-scale Least Squares Problems*. Report LIDS-P-2819. Cambridge: Laboratory for Information and Decision Systems, MIT. 2009.

[51] H. Yu, D. P. Bertsekas. Convergence results for some temporal difference methods based on least squares. *IEEE Transactions on Automation Control*, 2009, 54(7): 1515 – 1531.

[52] D. P. Bertsekas, J. N. Tsitsiklis. *Neuro-dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[53] H. Yao, Z. Liu. Preconditioned temporal difference learning. *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland. 2008: 1208 – 1215.

[54] D. P. Bertsekas, E. Gafni. Projection methods for variational inequalities with applications to the traffic assignment problem. *Mathmatical Programmming Studies*, 1982, 17(17): 139 – 159.

[55] B. Martinet. Regularisation d'inequations variationnelles par approximations successives. *Revue Francaise Informatique Recherche Operationnelle*, 1970, 4(R-3): 154 – 159.

[56] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 1976, 14(5): 877 – 898.

[57] D. P. Bertsekas. *Convex Optimization Theory*. Belmont, MA: Athena Scientific, 2009.

[58] G. Strang. *Linear Algebra and its Applications*. 4th ed. Wellesley, MA: Wellesley-Cambridge Press, 2009.

[59] L. N. Trefethen, D. Bau. *Numerical Linear Algebra*. Philadelphia: SIAM, 1997.

[60] J. A. Boyan. Technical update: least-squares temporal difference learning. *Machine Learning*, 2002, 49(2/3): 1 – 15.

[61] A. Nedić, D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems: Theory and Applications*, 2003, 13(1/2): 79 – 110.

[62] D. P. Bertsekas, V. S. Borkar, A. Nedić. Improved temporal difference methods with linear function approximation. *Learning and Approximate Dynamic Programming*. J. Si, A. Barto, W. Powell, et al., eds. New York: IEEE, 2004: 231 – 255.

[63] D. S. Choi, B. Van Roy. A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems: Theory and Applications*, 2006, 16(2): 207 – 239.

[64] J. Liu. *Monte Carlo Strategies in Scientific Computing*. New York: Springer-Verlag, 2001.

[65] R. Y. Rubinstein, D. P. Kroese. *Simulation and the Monte Carlo Method*. 2nd ed. New York: John Wiley & Sons, 2008.

[66] N. Polydorides, M. Wang, D. P. Bertsekas. *Approximate Solution of Large-scale Linear Inverse Problems with Monte Carlo Simulation*. Report LIDS-P-2822. Cambridge: Laboratory for Information and Decision Systems, MIT, 2009.

[67] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge: Cambridge University Press, 2008.

[68] R. Munos. Error bounds for approximate policy iteration. *Proceedings of the 20th International Conference on Machine Learning*, Washington D. C., 2003: 560 – 567.

[69] D. P. Bertsekas. Pathologies of temporal difference methods in approximate dynamic programming. *Proceedings of IEEE Conference on Decision and Control*. New York: IEEE, 2010.

[70] H. Yu. *Least Squares Temporal Difference Methods: An Analysis Under General Conditions*. Report C-2010-39. Finland: Department of Computer Science, University of Helsinki, 2010.

[71] H. Yu. Convergence of least squares temporal difference methods under general conditions. *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010: 1207 – 1214.

[72] R. S. Sutton, C. Szepesvari, H. R. Maei. A convergent $\mathrm{O}(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 2008: 1 – 8.

[73] H. R. Maei, C. Szepesvari, S. Bhatnagar, et al. Convergent temporal-difference learning with arbitrary smooth function approximation. *Advances in Neural Information Processing Systems*, Vancouver, BC, Canada, 2009: 1204 – 1212.

[74] R. S. Sutton, H. R. Maei, D. Precup, et al. Fast gradient-descent methods for temporal-difference learning with linear function approximation. *Proceedings of the 26th International Conference on Machine Learning*, New York, 2009: 1 – 8.

[75] D. P. Bertsekas, H. Yu. Q-Learning and enhanced policy iteration in discounted dynamic programming. Report LIDS-P-2831. Cambridge: Laboratory for Information and Decision Systems, MIT, 2010.

[76] J. N. Tsitsiklis, B. Van Roy. Optimal stopping of Markov processes: hilbert space theory, approximation algorithms, and an application to pricing financial derivatives. *IEEE Transactions on Automatic Control*, 1999, 44(10): 1840 – 1851.

[77] H. Yu, D. P. Bertsekas. *A Least Squares Q-learning Algorithm for Optimal Stopping Problems*. Report 2731. Cambridge: Labotary for Information and Decision Systems, MIT, 2006.

[78] M. G. Lagoudakis, R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 2003, 4: 1107 – 1149.

[79] T. Jung, D. Polani. Kernelizing LSPE(λ). *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, New York: IEEE, 2007: 338 – 345.

[80] L. Busoniu, D. Ernst, B. De Schutter, et al. Online least-squares policy iteration for reinforcement learning control. *American Control Conference*, New York: IEEE, 2010: 486 – 491.

[81] D. P. Bertsekas. *Lecture at NSF Workshop on Reinforcement Learning*. Harpers Ferry, New York, 1996.

[82] V. V. Desai, V. F. Farias, C. C. Moallemi. Approximate dynamic programming via a smoothed approximate linear program. *Advances in Neural Information Processing Systems 22*. Vancouver, BC, Canada, 2009: http://www.moallemi.com/ciamac/papers/salp-2009.pdf.

[83] V. F. Farias, B. Van Roy. Tetris: a study of randomized constraint sampling. *Probabilistic and Randomized Methods for Design Under Uncertainty*. G. Calafiore, F. Dabbene, eds. New York: Springer-Verlag, 2006: 189 – 202.

[84] S. Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems*. Vancouver, BC, Canada, 2002: 1531 – 1538.

[85] I. Szita, A. Lorinz. Learning tetris using the noisy cross-entropy method. *Neural Computation*, 2006, 18(12): 2936 – 2941.

[86] B. Van Roy. *Feature-Based Methods for Large Scale Dynamic Programming*. Report LIDS-TH-2289. Cambridge: Laboratory for Information and Decision Systems, MIT, 1995.

[87] P. T. de Boer, D. P. Kroese, S. Mannor, et al. A tutorial on the cross-entropy method. *Annals of Operations Research*, 2005, 134(1): 19 – 67.

[88] C. Thiery, B. Scherrer. Improvements on learning tetris with cross-entropy. *International Computer Games Association Journal*, 2009, 32(1): 23 – 33.

[89] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control*. New York: Van Nostrand, 1992: 493 – 525.

[90] P. J. Werbös. Neurocontrol and supervised learning: an overview and valuation. *Handbook of Intelligent Control*. D. A. White, D. A. Sofge, eds. New York: Van Nostrand, 1992: 65 – 89.

[91] L. C. Baird. *Advantage Updating*. Report WL-TR-93-1146. Wright-Patterson Air Force Base, OH: Wright Laboratory, 1993.

[92] M. E. Harmon, L. C. Baird, A. H. Klopf. Advantage updating applied to a differential game. G. Tesauro, D. S. Touretzky, T. K. Leen, eds. *Advances in Neural Information Processing Systems*, Denver, CO, 1994: 353 – 360.

[93] D. P. Bertsekas. Differential training of rollout policies. *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing*, Allerton Park, IL, 1997.

[94] E. V. Denardo. Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 1967, 9(2): 165 – 177.

[95] D. P. Bertsekas. Monotone mappings with application in dynamic programming. *SIAM Journal on Control and Optimization*, 1977, 15(3): 438 – 464.

[96] D. P. Bertsekas, S. E. Shreve. *Stochastic Optimal Control: the Discrete Time Case*. New York: Academic Press, 1978.

[97] L. S. Shapley. Stochastic games. *Proceedings of National Academy Sciences*, 1953, 39(10): 1095 – 1100.

[98] D. P. Bertsekas, H. Yu. Asynchronous distributed policy iteration in dynamic programming. *Proceedings of Allerton Conferrence on Information Sciences and Systems*, Allerton Park, IL, 2010.

[99] D. P. Bertsekas. Distributed dynamic programming. *IEEE Transactions on Automatic Control*, 1982, 27(3): 610 – 616.

Dimitri P. BERTSEKAS studied engineering at the National Technical University of Athens, Greece, obtained his M.S. degree in Electrical Engineering at the George Washington University, Washington D.C. in 1969, and his Ph.D. in System Science in 1971 at the Massachusetts Institute of Technology. He has held faculty positions with the Engineering-Economic Systems Department, Stanford University (1971 – 1974) and the Electrical Engineering Department of the University of Illinois, Urbana (1974 – 1979). Since 1979 he has been teaching at the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (MIT), where he is currently McAfee Professor of Engineering. He consults regularly with private industry and has held editorial positions in several journals. His research at MIT spans several fields, including optimization, control, large-scale computation, and data communication networks, and is closely tied to his teaching and book authoring activities. He has written numerous research papers, and fourteen books, several of which are used as textbooks in MIT classes.

He was awarded the INFORMS 1997 Prize for Research Excellence in the Interface Between Operations Research and Computer Science for his book 'Neuro-dynamic Programming' (coauthored with John Tsitsiklis), the 2000 Greek National Award for Operations Research, the 2001 ACC John R. Ragazzini Education Award, and the 2009 INFORMS Expository Writing Award. In 2001, he was elected to the United States National Academy of Engineering.